6-15-2018

Applying Block Chain Technologies to Digital Voting Algorithms

Vishanth Iyer
Santa Clara University, vaiyer@scu.edu

Nathan Kerr
Santa Clara University, nkerr@scu.edu

Justin Meeken
Santa Clara University, jmeeken@scu.edu

Alex Seto
Santa Clara University, aseto@scu.edu

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Date: June 14, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Vishanth Iyer
Nathan Kerr
Justin Meeken
Alex Seto

ENTITLED

Applying Block Chain Technologies to Digital Voting Algorithms

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ~~ARTS~~ Science IN COMPUTER SCIENCE & ENGINEERING

yuhong xin

Thesis Advisor

D. Ling

Department Chair

Applying Block Chain Technologies to Digital Voting Algorithms

by

Vishanth Iyer
Nathan Kerr
Justin Meeken
Alex Seto

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science & Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 15, 2018

Applying Block Chain Technologies to Digital Voting Algorithms

Vishanth Iyer
Nathan Kerr
Justin Meeken
Alex Seto

Department of Computer Engineering
Santa Clara University
June 15, 2018

ABSTRACT

Voting is a fundamental aspect to democracy. Many countries have advanced voting systems in place, but many of these systems have issues behind them such as not being anonymous or verifiable. Additionally, most voting systems currently have a central authority in charge of counting votes, which can be prone to corruption. We propose a voting system which mitigates many of these issues. Our voting system attempts to provide decentralization, pseudo-anonymity, and verifiability. For our system, we have identified the requirements, implemented the backbone of the system, recognized some of its shortcomings, and proposed areas of future work on this voting system.

Acknowledgements

The authors wish to thank Dr. Yuhong Liu for her advisement on this project; Dr. Ed Schaefer for his continued feedback and support of our project as well as Santa Clara University for making this project possible in the first place. Finally, we would like to offer a special thanks to our parents, friends, and families for their dedication and support of our education here at Santa Clara University.

Table of Contents

List of Figures

List of Tables

0.1 Introduction

Since the advent of Democracy, the issue of how to elect and vote for leaders has been prevalent. In many democratic elections there is a lack of transparency into the voting system and many of these voting systems allow for the possibility of cheating the vote. Most voting systems involve machines where people can insert a card with their selected candidate and drop it in the machine for counting. This leaves the possibility for malicious agents to cheat the machine either through hacking or some form of social engineering. The public needs a more transparent voting system that is decentralized, anonymous, can be verified by anyone, and that makes it difficult to cheat or vote fraudulently. Additionally, voters should be able to track their own, and only their own, vote.

To that end, we will design an electronic voting system that is based on technologies that underscore current cryptocurrencies. The system will be decentralized, anonymous, and verifiable. This is unlike existing conventional solutions, which are centralized in voting locations and subject to human oversight. Moreover, such methods are not verifiable, that is to say, you cannot know if your vote was correctly registered.

In light of recent claims of electoral fraud, the proposed voting system will help not only developing nations with corrupt governments, but also all democratic nations in avoiding political corruption and with creating electoral transparency. We believe that our system will make the electoral system more visible, fair, and engaging.

0.1.1 Background

Since the advent of Democracy, the issue of how to elect and vote for leaders has been prevalent. In many democratic elections there is a lack of transparency into the voting system and many of these voting systems allow for the possibility of cheating the vote. Most voting systems involve machines where people can insert a card with their selected candidate and drop it in the machine for counting. This leaves the possibility for malicious agents to cheat the machine either through hacking or some form of social engineering. The public needs a more transparent voting system that is decentralized, anonymous, can be verified by anyone, and that makes it difficult to cheat or vote fraudulently. Additionally, voters should be able to track their own, and only their own, vote.

To that end, we will design an electronic voting system that is based on technologies that underscore current cryptocurrencies. The system will be decentralized, anonymous, and verifiable. This is unlike existing conventional solutions, which are centralized in voting locations and subject to human oversight. Moreover, such methods are not verifiable, that is to say, you cannot know if your vote was correctly registered.

In light of recent claims of electoral fraud, the proposed voting system will help not only developing nations with corrupt governments, but also all democratic nations in avoiding political corruption and with creating electoral transparency. We believe that our system will make the electoral system more visible, fair, and engaging.

0.2 Requirements

Before designing the Block Chain Voting System, a list of functional and non-functional requirements was created along with the design constraints.

0.2.1 Functional Requirements

- The system will be pseudo-anonymous - votes will be tied to keys but never to people.
- The system will minimize the need to trust a single authority.
- The system will be able to detect vote tampering.
- The system will allow voters to cast one vote and only one vote.
- The system will give a public ledger of votes so the total count can be verified by anyone.

0.2.2 Nonfunctional Requirements

- The system will be highly secure.
- The system will make it easy for users to cast and verify their votes.

0.2.3 Design Constraints

- The system will be able to run on any collection of machines with a connection to the Internet.

0.3 Use Cases

To achieve the requirements set out in the previous section, we have identified the following use cases for the two key actors in our system. The first voter is the Key Authority who is responsible for generating all public/private key pairs that will be used for the election. In practice, this will not be a single person and will likely consist of a decentralized and auditable ceremony attended by multiple experts in cryptography and election safety. For the purposes of this proposal however, we can think of the Key Authority as a single trusted entity that is able to generate and distribute keys in a secure manner.

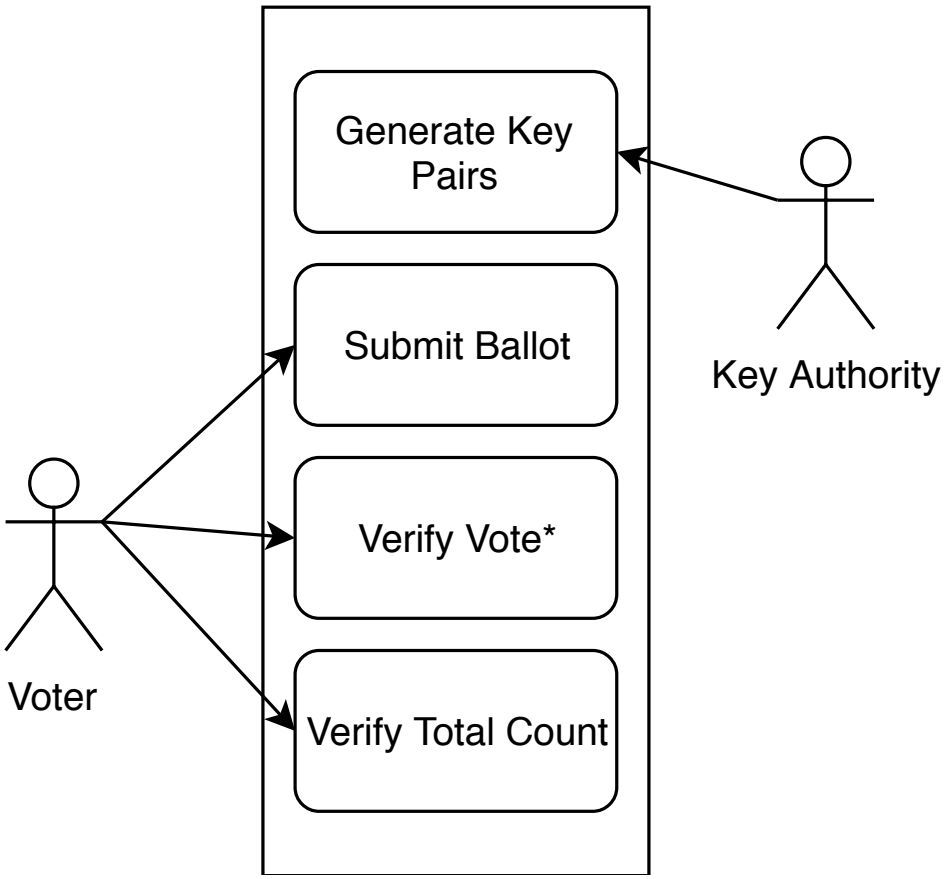

Figure 1: Use Case Diagram

0.3.1 Generate Key Pairs

Goal: Generate and distribute all key pairs that will be used in this election. Also compile a list of all public keys generated to give to the verification nodes.

Actor: Key Authority

Preconditions: Secure process has been prepared, known number of keys to generate.

Steps:

The exact steps needed to generate and distribute keys securely are out of scope for this project however research has been done into this process (see discussion on Corrupted Key Authority in Possible Vulnerabilities).

1. Get count of keys needed for this election.
2. Generate keys in a secure public ceremony.
3. Distribute keys randomly (so that no one knows which keys are tied to which locations).
4. Audit number of keys distributed to polling places to ensure that all keys are accounted for at the end of the election.

Postconditions: Keys have been generated and distributed to the polling centers.

Exceptions: Keys are lost or corrupted.

0.3.2 Submit a Vote

Goal: Allow voter to submit their ballot at one of the polling centers.

Actor: Voter

Preconditions: The actor is registered to vote

Steps:

1. Verify identity with volunteers at front and receive a random key pair.
2. Fill out a ballot at one of the voting machines.
3. Give key pair to the voting machine so that it can sign and submit the vote.
4. A subset of keys will then be wiped (more about this in Conceptual Model and the Coercion section of the Possible Vulnerabilities).

Postconditions: A vote has been submitted to the verification nodes for addition to the blockchain; a subset of the keys is wiped.

Exceptions: User tries to vote twice; voting machine is unable to contact the verification nodes.

0.3.3 Verify Own Vote was Counted

Goal: Allow a subset of voters to ensure their vote was counted by finding it on the Blockchain.

* Not all voters will be able to verify their vote after the election, only the small subset who did not have their keys wiped. This is to give protection against coercion.

Actor: Voter

Preconditions: Actor has voted; actor's key was not wiped after voting.

Steps:

1. Pull blockchain from one of the nodes (the blockchain is publicly readable).
2. Find their vote (identified by the public key) on the blockchain and check that the vote matches what they expect.

3. If the vote does not match, notify the election officials because it's likely that the voting machine itself was corrupted.

Postconditions: Voter will know that his or her vote was added to the blockchain successfully.

Exceptions: None

0.3.4 Verify Total Vote Count

Goal: Allow voters to ensure that the official tally was counted correctly.

Actor: Voter

Preconditions: Election has concluded.

Steps:

1. Pull blockchain from the nodes and take the copy that the majority agree upon.
2. Count the votes in the plaintext and verify that the final count is correct.

Postconditions: Voter will know total vote count is correct

Exceptions: None

0.4 Activity Diagrams

In our voting system, there is one actor: the voter. The voter can engage two different activities: either cast a vote or verify their vote was correctly cast.

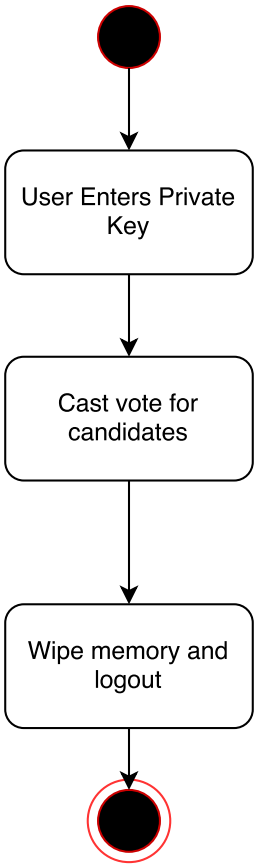

Figure 2: Voting Activity Diagram

Voters will start by 'entering' their private key, which will be stored in a flash drive and distributed to registered voters. The voter will then cast their vote for a particular candidate(s). After casting their vote, the user 'log out' and memory will be wiped to maintain anonymity.

To verify that an individual's vote was correctly cast, the user will use their private key to 'log in.' Then, they will be able to calculate the hash of the blockchain up until the end of their vote, and verify that value is the same as the one the stored in the blockchain - meaning the vote was correctly cast.

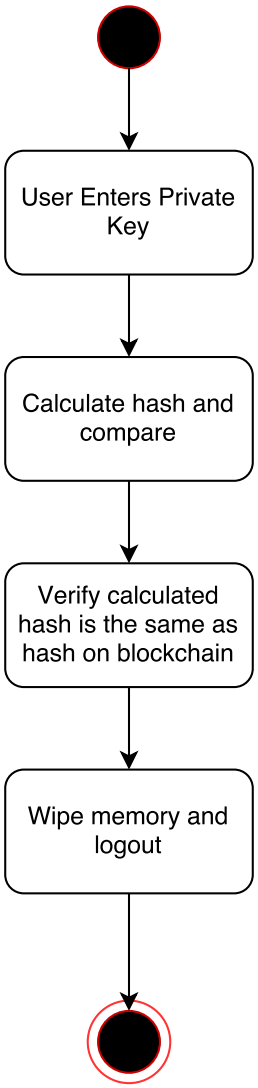

Figure 3: Verifying Activity Diagram

0.5 Conceptual Model

The voting process will look similar to current day elections. The voter will go to their polling place and verify their identity. They will then be given a random key from a collection that they can use to vote.

0.5.1 Overview of Voting Process

1. The key authority will generate public/private key pairs in a secure and public manner (see the Possible Vulnerabilities section for a discussion on how this might be done), the public/private keys will be put on a portable media and distributed to the voting centers. A list of the valid public keys will be sent to each of the verification nodes.
2. The user will verify their identity and be given a random public/private key pair from the collection.
3. The user then uses the key pair to vote at one of the voting terminals.
4. The user will then have the option to wait for their vote to be put on the blockchain and be given a confirmation from the voting machines.
5. At this point a predetermined percentage of the keys will be wiped and overwritten with the current timestamp (see discussion on coercion and duplicate voting in possible vulnerabilities). Refer to section on Possible Vulnerabilities for a more detailed usage of the timestamp.
6. The voters whose keys were not wiped will then take their keys home and be able to independently verify that their vote was added to the blockchain correctly.
7. At the end of the election, the verification nodes will be alerted that they should stop accepting votes. We then wait for a few more blocks to be solved to ensure that the nodes have reached a consensus about which votes should be in the blockchain.
8. Finally we pull the blockchains from the nodes and choose the longest one, we verify that the blockchain is legitimate and count the votes that appear in the plain text.

0.5.2 Counting and Verification

Any user can verify the blockchain themself because it is publicly readable. The user can go block by block to verify both that the signature of each vote is valid and that the block itself was solved correctly (see Architecture section).

Once the user is convinced of the validity of the blockchain, they can easily determine the election result because the ballots appear as plaintext in the blockchain.

0.6 Technologies Used

The system uses the following various technologies in the implementation of the web application:

- Python: Python is a general-purpose programming language known for its simple syntax and ease of use. Because our system is a proof of concept, we wish to use a language that will allow for quick development and easy incremental changes.
- Blockchain: A blockchain is a list of records that are linked and secured using cryptography. The blockchain will allow users to verify that their vote was counted while still preserving anonymity.
- Pycrypto: Pycrypto is a cryptography library for Python that implements many standard cryptography algorithms. Our implementation requires both a secure hashing algorithm and a digital signature algorithm. We will be using SHA256 and RSA respectively.

0.7 Architecture

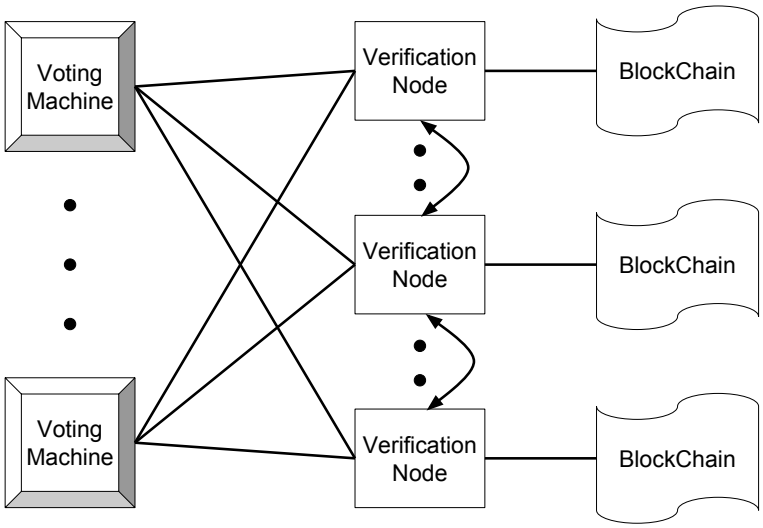

Figure 4: Architecture Diagram

Figure 4 is the Architecture diagram divided into 2 major sections: the voters (clients) and a network of verification nodes.

Public Key	Ballot (JSON)	Signature
MIGfMA0GCSqGSIb3DQEBAQU AA4GNADCBiQKBgQCqGKukO1 De7zhZj6+H0qtjTkVxwTCpv Ke4eCZ0...	{ "President" : { "John Smith": 0 "Suzy Q": 1 } "Senator": { "Jimmy McMillan": 1 "Malik Obama": 0 } }	wVQZKjeGcjDOL5UlsuusFn cCzWBQ7RKNUSesmQRMSGkV b1/3j+skZ6UtW+5u09lHNs j6tQ5...

Figure 5: Visualization of a Vote

0.7.1 Anatomy of a Vote

We will refer to the "ballot" as the user's actual selection of candidates and the "vote" as the combination of the user's public key, ballot, and signature (see fig. 5). The signature in the vote will be computed using the user's private key with the RSA algorithm.

0.7.2 Voting Machine

The voters will vote on special machines that take two inputs, the users' public and private keys, and store them in main memory to use for signing and hashing. The voters will then select their choices on the ballot. After selection is completed, the ballot will be signed using RSA. The voting machine will package the public key, ballot, and digital signature into a "vote" and send the vote to each verification node. The voter will have the option to wait at the terminal for confirmation that the block containing their vote has been added to the blockchain. We will consider a vote as securely on the blockchain when it is 4-5 nodes deep to ensure that the majority of nodes have adopted that particular chain. The verification process should only take 30 seconds to 1 minute.

0.7.3 Verification Node

The verification nodes are a network of machines that listen for votes, verify said votes are legitimate, and attempt to add those votes to their copy of the blockchain. Nodes will have a list of all valid public keys for votes to ensure each vote that arrives is legitimate.

When a node receives a vote, it will first verify that the digital signature on the vote is valid before attempting to add the vote to the block. If a vote is invalid because the ballot is malformed, the signature is incorrect, or the public key has already been used, the vote will not be added to the block and the voting machine will be alerted of the failure. If the vote is valid, the node will add the vote to the current block. To add a block into the blockchain, the verification node will continually hash the block until it solves a computationally expensive problem. The problem we will use the same as the proof-of-work used in the bitcoin protocol, namely looking for a hash that begins with a predetermined number of zero bits. This is implemented by incrementing the nonce inside the block until a value of the nonce is found that gives the block's hash the required zero bits [4]. Votes will be continually added to the current block as the node receives them until the block is successfully solved, at which point the node will begin adding incoming votes to a new block and attempt to solve the new block.

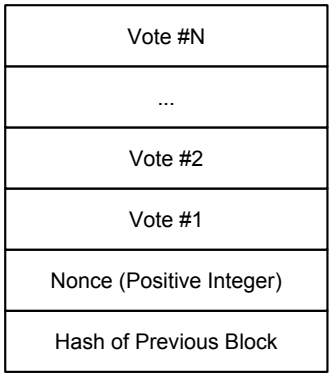

Figure 6: Visualization of a Block

After the node has solved the problem (found an appropriate hash), it will broadcast the solved block to the rest of the verification nodes. Each node will check the signatures of the votes to ensure they are valid and that the problem was solved correctly and, if successful, add the new block to their respective copy of the blockchain. To check if a signature valid, the node will use a digital signature algorithm - in this case, RSA - to verify that the ballot was cast by the individual whose public key appears on the vote. If any vote is invalid, the node will reject block. If all votes are valid, the node will check that the block was solved correctly by hashing the block and making sure the block's hash has the required number of zero bits. Verifying that the block was solved correctly only requires the computation of a single hash. Once the verification node checks each vote in a block is valid and that the block has been solved, it will add the block to its copy of the blockchain and mark the public keys in each vote as 'used' in its list of authorized

public keys. The node will then start a new block with the remaining votes that were not contained in the previous block and begin solving the new block.

0.8 Design Rationale

In this section, we explain why we chose our design and technologies that we will plan to use.

0.8.1 Design Justification

Here are the essential features that we addressed in our design:

- On the surface, our system may appear fairly simple. Our use cases are simply voting and verifying the votes. Since we are using a decentralized architecture, all voters in our system are more or less equivalent, with exception to the possible randomized subset of users that will be provided keys to verify their own vote. Additionally, becoming a verification node that is capable of adding to and modifying the blockchain will be limited to government entities. We assume that since there are so many government entities with conflicting interests, it will be difficult to come to a consensus between the entities to get a majority control over the blockchain. In practice, there would likely be community leaders and technical experts that will have more in depth knowledge of the system but they will have the same privileges as everyone else. This ensures that no single person or group has more power in the voting process.
- For similar reasons as above, we will be using P2P (Peer to Peer) architecture. This will allow government entities to set up their own nodes and help contribute to verifying that the election was legitimate. Additionally, a subset of users will be able to take their keys home so they can download the blockchain to ensure their vote was counted.
- The use of homomorphic encryption will allow the system to sum votes and return their count. For example, if we have a key S_k, and 4 votes V_1, V_2, V_3, and V_4, we can encrypt the votes as $S_k(V_1)$, $S_k(V_2)$, $S_k(V_3)$, and $S_k(V_4)$ with the property that $S_k(V_1) + S_k(V_2) + S_k(V_3) + S_k(V_4) = S_k(V_1 + V_2 + V_3 + V_4)$. This has the implication that we can verify a sum total of votes without knowing what any single individual voted. The notion of multi-key homomorphic encryption means that we can have each use encrypt their votes independently.

0.8.2 Justification for Technologies Used

We will be using the following technologies for our project:

- Python: As previously stated, Python allows for rapid development. Because our project is heavily based in theory, we want to be able to quickly implement changes to test their feasibility and may need to change details of our implementation quickly to adjust for new ideas and understanding. Additionally, Python already has secure implementations of many needed cryptography packages which will allow for us to avoid the need to re implement any already implemented cryptography protocols that will be used in our design such as hashing, random number generation, and public key cryptography. The nature of Python and our previous experience with the language makes it the best fit for our needs.
- Basic Cryptography Technologies: We will need many basic cryptography packages such as a secure random number generator, a secure hashing algorithm like SHA256, and public key cryptography, specifically RSA. The combination of these technologies will allow for secure generation of keys, signing of votes to verify authenticity, and will provide the backbone needed to create a blockchain.
- Blockchain: The core of our project is centered around creating a verifiable but anonymous record of all the votes cast. We will do this by creating a blockchain, chaining the hashes of the votes. The blockchain allows the user to verify that their vote was actually counted and reassures the voters that the system is fair. Additionally, blockchain decentralizes the verification and vote counting process, so in our case multiple authorities will be verifying votes rather than just one authority. This allows our system to avoid putting trust in a single authority

to count the votes correctly. In our system, government entities or agencies will be able to sign up to become verification nodes. It is assumed that since each of these agencies or entities likely have different or conflicting interests they will not be able to collaborate to corrupt the verification process as a blockchain based system would require a majority consensus to override or corrupt individuals votes.

0.9 Possible Vulnerabilities

When creating a system dealing with sensitive data, it is important to consider the possible security vulnerabilities and the ways they could be exploited. In the following section, we discuss security issues and mitigations.

0.9.1 Coercion

The importance of protecting against coercion will differ widely between use cases. In more stable democracies and less important elections coercion may not be a serious concern while in other situations it may put people at significant risk. It is very difficult to reconcile verifiability with resistance to coercion, and true anonymity.

Our system could be hardened against coercion by wiping each key after it is used to vote so that no person can be forced to reveal which vote was theirs. This will however preclude them from verifying their vote independently. Instead they will need to wait at the voting terminal for confirmation that their vote was added to the blockchain. It also will allow the possibility of a corrupted voting machine sending the wrong vote and faking a confirmation.

At the other extreme, if we allow each user to keep their key and verify their vote manually, we can ensure that each vote was correct and counted. This comes at a steep cost however as it now becomes much easier to coerce someone into revealing their vote because they can be forced to turn over their key storage device.

We chose to take a compromise between these two approaches. Depending on the size of the election, the key authority will set a percentage of the keys (say 1/100) that will not be wiped after use. This allows some number of voters to verify that their votes were counted correctly, while also giving voters plausible deniability by giving them the option claim their key was wiped in the case of coercion. The intact keys will provide some measure of defense against corrupted voting machines by exposing vote manipulation.

0.9.2 Duplicate Voting

We will define duplicate voting as one voter voting multiple times. Within this we see two possible scenarios, in the first the voter tries to use their key to vote again. This is easily handled because the verification nodes will keep track of the valid public keys and mark them as used once a vote is received. So, even if the key is not wiped, it cannot be used to vote again. Second, and slightly more difficult to mitigate against, is the case where a user successfully votes and then claims that their key did not work.

If the key was corrupted, the machine will reject it without wiping it and we can confirm that the voter really did not get to vote. Assuming the key was wiped, we cannot verify whether this key was indeed added to the blockchain and are unable to tell if the voter is lying or not. To prevent this, the voting machine will write the current timestamp to the key after wiping it. This way, if a voter claims that their key did not work but the timestamp signals that the key was wiped recently, it is likely that they are lying. If, however the timestamp is from a while ago, it is possible that the keys were compromised and the authorities should be alerted so they can investigate.

0.9.3 Corrupted Voting Machine

The voter must place a certain level of trust in the voting machine. Since the voting machine has access to the private keys, it is possible that the machine can cast a vote that is different from what the voter intended and then display that the vote was added correctly. In the cases where the key is wiped, there will be no way the voter will know his or her vote had been tampered with. However, when allowing a percentage of users to keep their keys, the vote tampering will be apparent and provable when the subset of verification users go to verify their votes.

0.9.4 Corrupted Verification Node

If one or more verification nodes are compromised, then there is an opportunity for an attacker to place fake votes into a block, verify the block, and broadcast the block to the other nodes in the blockchain. However, it is assumed that a majority of the nodes in the network will be "honest" nodes. Under this assumption, the honest nodes will have more computing power than the compromised nodes, and thus faking a block or a portion of the blockchain is highly unlikely. Furthermore, faking an individual block is impossible because each vote is digitally signed by the voter's private key, and since the compromised node will not have access to the voter's private key, it will not be able to fake a vote. Thus all blocks with fake votes will be rejected and never placed on the blockchain.

0.9.5 Corrupted Key Authority

Unfortunately, most voting schemes will require some trusted party to oversee the election. In our system, most of the "blind trust is centered around the vote authority. There are a few possible vulnerabilities in the key authority that cannot be mitigated within our system. For example, the key authority may leak private keys which could be used to add illegitimate votes to the blockchain. Or, if the key authority keeps track of which keys are distributed to various locations, it harms the anonymity of our system (for example we know that someone in a small town voted at 4 p.m. and only these keys went to that location, so we can link the votes in the blockchain to individuals). Finally, if the keys are not generated in a cryptographically secure manner, it may be possible for someone to generate their own keys.

While our system itself does not provide any mitigations for these attacks, in practice there are ways to minimize the risk. For example, public ceremonies are often used to generate keys in a secure manner. [3], practical example, https://www.iana.org/dnssec/ceremonies/22) In a typical ceremony, every person present must be identified and cleared, the system being used will be validated by multiple third-party experts, every command that is entered will be carefully recorded and verified, the system itself may be in a locked vault for the duration of the key generation, and finally the entire process is recorded so anyone can verify that there was no manipulation. We believe that mitigations such as the public ceremony will be sufficient to protect against any major attacks on the key authority.

0.10 Test Plan

In addition to standard unit testing to verify good coding practices, we will have a battery of tests to address each of our major system requirements.

- The system will be anonymous for each voter.

  We will perform multiple types of attacks on the system to determine any weak points. For example, we attempt to perform a man in the middle attack wherein we attempt to determine the identity of the voter or the actual vote itself.

- The system will be decentralized.

  We will disable several of our endpoints during testing to ensure that our system remains available despite missing servers.

- The system will allow users to verify their own vote (any only their vote) was counted.

  We will be testing the vote verification system to determine if as a voter we can verify the status of our vote, but that no other individual can do the same.

- The system will allow voters to cast one vote and only one vote.

  We will attempt to cast multiple votes on our system, both as a normal user, and as malicious actor. As the latter, we will try everything we can to create many false votes.

We will be performing continuous testing throughout the entire process to ensure that our system is hardened against any such attacks. We believe we will be able to anticipate and prevent most, if not all, attack types.

0.11 Risk Analysis

It is important to analyze risks to correctly estimate the project's progress. To do so the potential risks were tabulated along with the probability and severity, with a ranking between 1 and 10. These two were multiplied to get the risk impact. Along with these calculations there are mitigation strategies to avoid these risks.

Risk	Consequences	Probability	Severity	Impact	Mitigation
Poor Team Coordination	The team will not finish the project or will not be able to meet deadlines.	0.7	8	5.6	Make clear deadlines and communicate with team.
Illness	Team member will be unable to contribute for period of time.	0.6	3	1.8	Ensure that all team members are familiar with all aspects of the system.
Bugs	Irregularity in systems performance and time wasted debugging.	0.5	5	2.5	Thoroughly understand and test the implementation.
Insufficient Knowledge of Blockchain	Extra time spent designing system or incomplete system.	0.15	10	1.5	Spend time researching and understanding blockchain before design.
Poor File Management	Lost work and time wasted trying to redo work.	0.02	10	0.2	Maintain a copy of current project, use Git, and backup files.

Table 1: Risk Analyzer

0.12 Development Timeline

A development time line was created and organized into the following subsections: requirements, design, implementation, testing, and documentation. For each task, a member of the group is assigned a timeline labeled by cell colors. The legend has the timeline labeled by the colors of the corresponding group members of the team and the deadline is indicated by a red border.

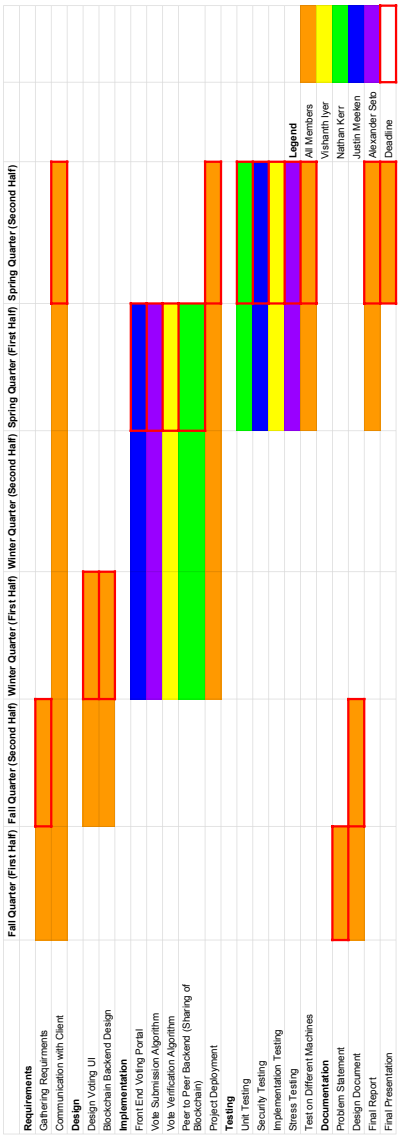

Figure 7: Project Timeline

0.13 Social Analysis

When creating a voting system used on a massive scale by many people, it is important to consider the many societal impacts it may have. Our system has a major impact in political, social, and ethical areas.

0.13.1 Ethical

Many ethical questions came up in the development of our voting system. One that we particularly focused on was how to prevent coercion and anonymity in a voting system which provides proof of how one voted. It is important to ensure that each vote is private and was not influenced by another party aside from the voting individual. This voting system provides a means of plausible deniability to provide protection from coercion and corruption. The system accomplishes this with the proof of vote deleting mechanism discussed early to provide protection to verifiers by giving them the option to deny the proof of their vote in the case they were asked to show this by a coercer. Throughout the development of this voting system it was a priority to protect the integrity of every single vote and to not allow for corruption in the overall voting process to make this an ethical voting system.

0.13.2 Social

This project should have a major social impact similar to the way it has a major political impact. It will hopefully transform the way people vote and allow for a more democratic society. It should be able to accomplish this by allowing people to vote more freely without the fear of the vote being revealed or manipulated. In doing this, it should also increase civic engagement and participation by the general public allowing for a more vocal and engaged society in political matters.

0.13.3 Political

This project will have a major political impact in that it could entirely change the way people vote. People will have the benefit of being able to vote as they wish with no risk of their vote being exposed and will be able to also ensure that their vote matters. Because people will know when their votes have been counted in this voting system, it will hopefully encourage a more civically engaged public.

0.13.4 Economic

This project should be very low cost for development. As it is aimed to be open source, development will be open to anyone to help maintain and ensure the system functions properly. Overall, the only cost will be the small amount of development time put in by contributors to this project. Otherwise, the costs of the voting machines and key distribution media are up to the government or organization that chooses to implement this voting system.

0.13.5 Health and Safety

As the process of voting itself does not concern one's health, this section is not relevant to our project.

0.13.6 Manufacturability

This voting system should be easy to manufacture and distribute as it could be designed to be compatible with current voting booths. It could simply be a matter of a software update to distribute this system to voting machines.

0.13.7 Sustainability

As this is an entirely software based voting system, it should have little impact on the sustainability of the voting machines themselves.

0.13.8 Environmental Impact

Only the voting machines themselves and the key distribution mechanisms themselves will have environmental impacts, which are both out of the scope of this project as this is entirely a software project.

0.13.9 Usability

One concern we have with our voting system is making it usable for anyone. If the voting system were to not be easily usable, it could have the impact of not giving certain people a voice in a democracy because the barrier to vote is too high. With our system, we intentionally keep the voting portals simple and familiar so that anyone who is currently able to vote will have no more difficulty voting on one of our portals.

0.13.10 Lifelong Learning

This project helped our group prepare for working in the industry, where we will need to pick up new technologies, which were not discussed in class, fairly quickly. In this project we had to learn how block chain works and how to apply it to voting. This ability to pick up complex technologies quickly and apply them to new areas will be something we will carry with us to our future careers.

0.13.11 Compassion

This project is compassionate in that it allows those without a voice in current political systems to be heard. This includes political systems that are prone to corruption and coercion where votes could be counted improperly or people could be forced to vote a particular way. This attempts to reduce, and hopefully eliminate, these problems by providing anonymity and transparency. By providing each of these, people in corruption prone locations can have the security of knowing their votes were counted and were kept anonymous.

0.14 Conclusion

We have proposed a system for secure digital voting that has three key traits: decentralized, pseudo-anonymous, and verifiable. The voting system uses current voting infrastructure and is familiar enough such that voters will not have to change the way they cast their ballots. With the use a blockchain as a distributed datastore, we can guarantee that votes cannot be tampered with. Using public key cryptography, we are able to guarantee pseudo-anonymity as well as verify that the ballot was cast as intended. We have also outlined certain attacks on the system that may compromise the aforementioned key traits and we have discussed potential mitigations to each of the attacks.

0.14.1 Lessons Learned

- The voting system is complex. There are many things that need to be taken into account when designing a robust system that can work in various political scenarios.
- "Secure" systems need to work correctly all the time. If there is one component that does not work as intended, or there is some inherent flaw with that component, the whole system can be compromised. The security of the system is predicated on the security of its parts.
- Good team work is not always easy. It was difficult at times to coordinate schedules and meetings. Accountability was also something we could have done better on. Sticking to deadlines would have helped

0.14.2 Future Work

Future work on this project involves implementing some of the subcomponents. The implementation of wiping a subset of keys needs to be completed to give the system coercion resistance. This involves finding a mechanism that is able to wipe keys and not directly connected to voting machines or verification nodes. Determining key distribution media is also important when deploying the full voting system. This aspect was out of the scope for our current project, but it is an important consideration. Furthermore, the design of the actual voting terminal needs to be completed. Again, this was out of the project's scope, as the entire terminal must be a secure system.

0.15 References

Bibliography

[1] A Barnes, C Brake, and T Perry. Digital voting with the use of blockchain technology. *Team Plymouth Pioneers–Plymouth University*, 2015.

[2] Josh Benaloh. Rethinking voter coercion: The realities imposed by technology. *The USENIX Journal of Election Technology and Systems*, 82, 2013.

[3] Carl Ellison. Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399, 2007. https://eprint.iacr.org/2007/399.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf.

[5] Ryan Osgood. The future of democracy: Blockchain voting'. *COMP116: Information Security*, 2016.

[6] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, May 2014.