Eötvös Loránd University

Faculty of Informatics

Department of Computer Algebra

CryptoVote: Blockchain based transparent election protocol

Supervisor:

Péter Ligeti

Department of Computer Algebra

Submitted by:

Valeh Hajiyev

Computer Science, MSc.

Budapest, 2018

ACKNOWLEDGEMENTS

I would like to thank Prof. Peter Ligeti, for his logistic support and valued advice which encouraged me to furnish this master thesis, without which the thesis would not have seen the light of the day.

I am thankful to Eötvös Loránd University for giving me wonderful opportunity to study and learn in an excellent environment.

Nevertheless, I am also grateful to Mr. Zsolt Borsi for his patience and help in overcoming numerous obstacles I have been facing through my master studies.

Last but not least, I would also like to thank my faculty members, family and friends for their support and encouragement.

Thank you all.

SUMMARY

From doing shopping, ordering lunch to paying bills and filing tax return, most of the important transactions in our life can now be carried out online, but one crucial process is still stuck in the dark ages: voting. In today’s society where, electronic technology is growing at an ever-increasing rate, governments are still using paper-based voting systems. These systems require a person to go to a particular place at a particular time on a particular day and fill out a piece of paper that will be processed along with thousands of other pieces of paper by a handful human beings. This process can be quite error-prone, non-transparent, and can lead to fraud and cheating in corrupt governments.

E-voting offers a possibility to improve democracy to get stronger voice over very large number of people by making the procedure easier. Unfortunately, a majority of e-voting systems are backed by central databases which themselves are controlled by central authorities. In such systems, possibility of someone, either the central authority or a hacker to make changes to the records in the centralized database is quite high. Therefore, the e-voting systems which are backed by centralized databases are not reliable either.

Blockchain technology, the technology that underlies crypto currencies such as Bitcoin is the ultimate solution to improve the voting process. Blockchain provides a decentralized, peer-to-peer and immutable database. Data on the blockchain is copied on every node that is a part of the network. Instead of the central body, the network decides whether a particular transaction should be recorded or not. This is solved by using consensus amongst all nodes on the blockchain. If a new record is added once on the blockchain, it cannot be reversed. There are no practical ways of going back and changing or deleting any data that has been recorded on the blockchain. These features make blockchain very strong technology underlier for e-voting systems.

In my thesis, I am going to design an election system which is built on top of blockchain technology.

TABLE OF CONTENTS

1. ELECTRONIC VOTING

1.1 Overview

Electronic voting, in particular voting through the Internet, is a discussion subject in many countries at the moment. In spite of the fact that some e-voting models are already widely used by public organizations, in the private sector, for informal and consultative polls, the situation looks different in the case of nationwide elections and referendums. Some countries plan to introduce electronic voting and organize a number of experiments. In some countries, due to information security problems, electronic voting, in particular the use of the Internet in elections and referendums encounters struggle [1].

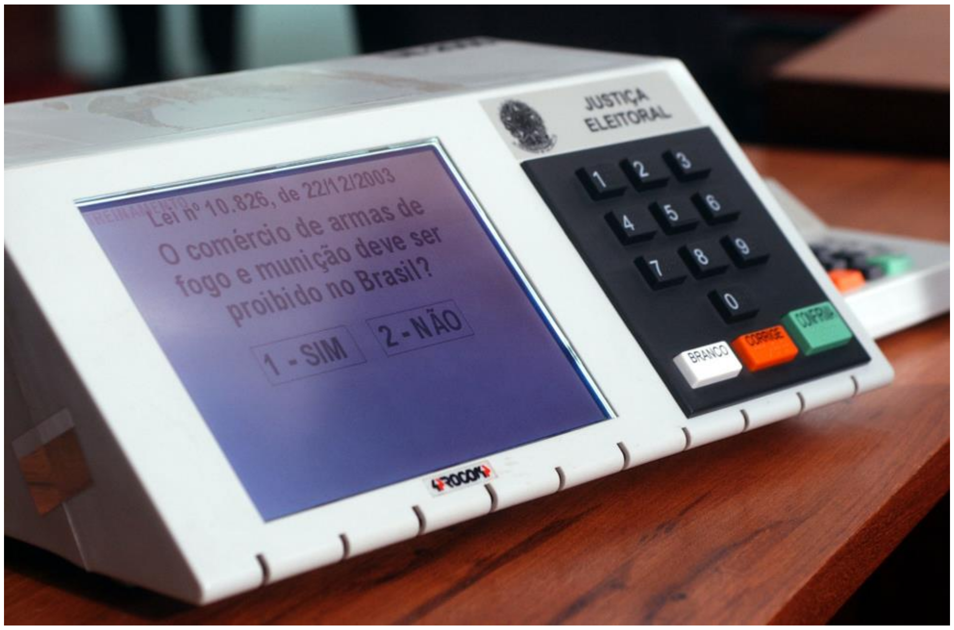
JUSTIÇA ELEITORAL
Lei nº 10.826, de 22/12/2003
O comércio de armas de fogo e munição deve ser proibido no Brasil?
1 - SIM
2 - NÃO
BRANCO
CORRIGE
CONFIRMA

Figure 1: Direct recording voting machine developed in Brazil and used in Brazilian elections

Source: José Cruz – AgenciaBrasil

What is e-voting?

The term "electronic voting" designates use of electronic means of vote on elections and referendums. There are several electronic voting systems. For example, direct-recording electronic voting systems (DREs) that register votes in the device memory without transmitting data on the Internet or any other network. Optical scan voting systems read the mark delivered by the voter on the ballot [2]. There is also an internet voting in which a personal computer with an Internet connection receives the voter's mark and transfers it to a different, remote computer in order to store it. Most devices with the Internet connection including tablets and smartphones can also be used for electronic voting.

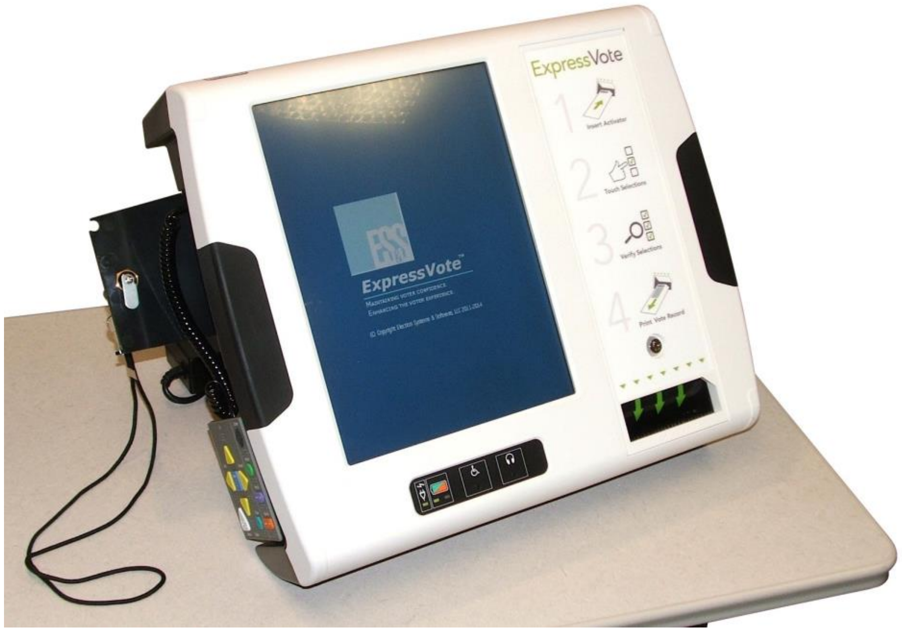

Figure 2: ExpressVote – an electronic ballot marker, made by Election Systems & Software. This prints a narrow ballot containing a summary of votes cast in both human-readable and bar code form.

Source: Douglas W. Jones

Electronic voting can be organized in various ways: stationary based – using electronic voting devices within the polling stations, or remote – by voting outside the polling stations (from home, workplace, etc.) via a personal computer through the Internet. Both types of e-voting face similar problems. However, some aspects of remote e-voting are more problematic than e-voting at polling stations. Additional difficulties in the case of remote e-voting are the lack of control over the process outside the polling stations and the need of transmitting data about the votes cast to the vote counting centers.

1.2 Electronic voting requirements

Traditional election systems are designed to satisfy the principles of democratic elections and referendums, to be specific, to ensure the free expression of will and anonymous voting, to prevent coercion or influence on voters, and to guarantee the integrity of election results. It is important to comply with all these principles when introducing new methods of voting. The electronic voting system has to be designed in such a way so that to ensure the reliability and safety of the voting process.

E-voting has to be free, transparent, anonymous, reliable and secure mechanism.

Thus, the e-voting system must meet the following minimum requirements:

1. Provide that only eligible voters can vote;
2. Guarantee transparent counting of each given vote and the given single vote has to be counted only once;
3. To satisfy the voter's right to form and express his opinion freely, without any coercion or influence;
4. Guarantee the privacy of the vote at all stages of voting;
5. Provide access to the system to the widest possible range of voters;
6. Strengthen the confidence of voters by providing comprehensive information on the functionality of the system.

1.3 Audit of e-voting systems

Similar to traditional voting methods, e-voting should provide the possibility of auditing - checking the process of obtaining and counting votes and recounting votes in order to confirm the correctness of election results. The most serious threat to e-voting systems is the risk of unauthorized intervention in the system, which, if not detected, leads to invalid voting results. Therefore, independent and comprehensive monitoring of information security, audit, cross-checking and timely reporting should be important components of e-voting systems.

There are various mechanisms for auditing electronic voting systems. Some systems include confirmation about the vote on paper such as "voter-verified paper audit trail" (VVPAT). VVPAT issues a confirmation paper to the voter at the time of voting in order to confirm their vote before the final submission [3]. These printed paper bulletins can also be used to recount votes later. This method can only be used for stationary electronic voting, since the voter must be physically present. In addition, audit of e-voting systems may include making e-voting software open source so that voters, political party representatives and civil society can study them.

Regardless of which e-voting audit method is used, it is very important that audit is available in each stage of the voting process (registration, vote, counting). The audit system should also provide independent observers to the system. In general, the e-voting audit has to be designed to detect any fraud case in the election and provide confirmation that all counted votes are valid.

Audit systems, by their nature, collect a set of various inputs. However, storing too much information may lead to privacy leaks, and the secrecy of the vote might be violation. Therefore, the audit system should ensure the preservation of the voter's anonymity during the all stages of the voting process. Data collected by the audit system should be protected from unauthorized access in all situations.

1.4 Advantages and risks of e-voting

Supporters and opponents of e-voting provide various arguments. On the one hand, the Internet based remote e-voting technologies:

- Attracts a larger number of voters to vote;
- Allows voters to vote outside the polling station in which they are registered and offers an alternative to voters who previously voted by mail;
- Reduce the overall costs of organizing and conducting the election process over time;
- Allow the counting of votes and announcing the results of elections in a shorter period.

Potential difficulties with e-voting

- Unauthorized interference in the election process of third parties. Taking into account the current level of information technology development, it cannot be guaranteed that the software will not be subject to manipulations that allow storing or printing forms other than those displayed on the screen.
- Unlike the paper-based solution, it is more difficult to identify a source of malfunctions and failures of the equipment.
- The possibility that in case of failure of a fully automated system and the absence of the backup data, the recount of votes will be extremely problematic or completely not possible at all.

1.5 E-voting challenges

In the context of remote e-voting, special attention should be given to the process of ensuring the privacy of voters. Access to the voting system should be granted only to citizens who have the right to vote; which means that every voter must be authenticated (for example, using digital signature) and his or her voting right must be confirmed. To prevent duplication of the votes and other violations, it is necessary to keep the record of the votes. Data should be stored in a way that establishing any connection between the vote cast and the particular voter should not be possible. The election system has to be fault tolerant. In order to make it transparent, election results should be open to everybody.

Most of the existing election systems does not meet the requirements above. We believe using the ring signatures and blockchain – the underlying technology of crypto currencies, we can introduce a new transparent, decentralized and durable election system that meets the requirements above.

2 BLOCKCHAIN

2.1 Overview

The first attempt to solve the problem of secure payments on the Internet was made in the early 1980s [4]. The main challenges were that the user had to report too much information about himself, and there was always a third party. This means that there was a counterparty risk and the need to pay a commission. Later in the beginning of the 1990s, the first digital payment systems started to appear, which were relatively safe and allowed anonymous online payments (MilliCent, DigiCash, PayPal) [5]. However, several years later most of those companies went bankrupt - then users were concerned about the security and privacy of payments online.

In 2008, Satoshi Nakamoto[1] published a paper, which describes a new protocol of electronic payment systems – Bitcoin [6]. Bitcoin is a form of digital currency that is created and held entirely electronically instead of being printed currency. Unlike banks and traditional fare currencies, no one controls or runs Bitcoin. The Bitcoin algorithms control the rate at which new Bitcoins are introduced, and people are incentivized to help maintain and mine new coins by being paid themselves in Bitcoin. Bitcoin is created for the peer-to-peer network. The peer-to-peer network is based on the equality of participants. It does not have dedicated servers, and each node (peer) acts as both a client and a server. Soon the first application for maintaining and using the bitcoin network was released. The network began to rapidly gain followers, and Bitcoin started to become more popular in the global financial industry [7]. Bitcoin was a first in a new category of currencies called cryptocurrencies, where people could exchange money without third parties online. Bitcoins can be spent electronically for both digital and physical goods. There are even vending machines that lets people to purchase goods by using your mobile phone to pay for products with Bitcoin. The most important feature of Bitcoin is that it is entirely decentralized [8]. No one company or government owns or has any control over Bitcoin. The decentralization in Bitcoin achieved by its underlying technology – blockchain.

[1] Satoshi Nakamoto is an assumed identity and today nobody really knows who he is, although there have been many claims. There is also a theory that Nakamoto is a group of people.

What is blockchain?

Blockchain is a digital public ledger that keeps the records of transactions in a public or private peer-to-peer network. Distributed among all nodes of the network, the ledger continuously records the history of operations with assets in the chronological order in the form of blocks of information. Instead of relying on the third parties, the nodes of the blockchain network use a special consensus protocol to maintain the content of the ledger, as well as cryptographic hash algorithms and digital signatures to guarantee the integrity of the transactions and the transfer of assets [8].

At the center of the blockchain there is a record of transactions much like a traditional accounting ledger. These transactions could be a movement of money between people or companies, or it could be any piece of information that is transactional, like the transferring of property deeds or the tracking of movement of inventory between different companies. Data stored in a blockchain is designed to be kept in a way that makes it virtually impossible to change the data once it is in the blockchain without being detected by other users. Traditional banking transactions are verified by a central bank or authority, while blockchain applications could replace these more centralized systems with these centralized ones where verification comes from the consensus of multiple users participating in the blockchain.

The consensus mechanism guarantees that distributed ledger among the peers are exact copies, which reduces the risk of fraudulent transactions. Cryptographic hash functions, such as the SHA256 hashing function, ensure that any change in the input data of the transaction, even the most insignificant one, will result in a different hash value in the block checksum, which is the indicator of the corrupted transaction. Electronic digital signatures guarantee that transactions are performed by legitimate senders, which means the transactions are signed by the corresponding private key of the public key of the sender.

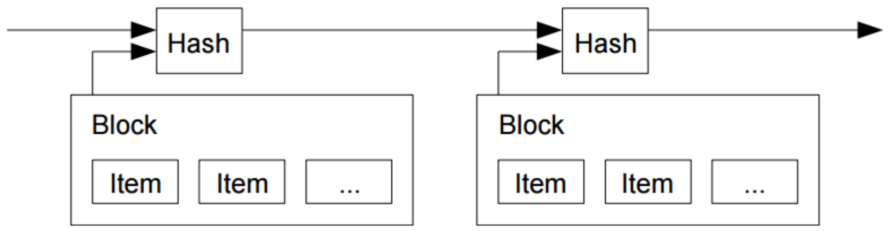

Figure 3: A visual illustration of blockchain as a data structure.

Source: Bitcoin: A Peer-to-Peer Electronic Cash System, S. Nakamoto

A blockchain has to do two main things. It needs to gather data or transactions together and put them in blocks. Then those block needs to be chained together securely using cryptography. When the transactions are put onto the block, we use cryptographic hashing or digital fingerprinting to link the transactions together. This means that if any part of the transaction changes, the entire block will fail verification, which will flag it to the other users.

At the top of the block, we have a hash that represents all of the transactions in that block. It is called a block address. Block addresses are used to chain the blocks together because each block knows who his previous block address is, which is how the chain is formed. For a block to be entered in the chain, the peer also known as a miner creating the block has to solve a complex mathematical puzzle when calculating the block address. This is called mining the block with proof-of-work. The puzzle was designed to be computationally expensive and has to happen before each block in the chain.

Let's assume this puzzle to calculate the block address takes 10 minutes to solve. If we have 1000 blocks in our chain, the total time spent calculating all the block addresses is 166 hours. If someone wants to overwrite the data in the first block, they have to recalculate that block's mining puzzle, which takes 10 minutes. But because all of the other blocks are cryptographically linked, if you change a block, then you need to recalculate the hashes and addresses for the next block and so on and so on. In order to change data in block 1 means that 166 hours or nearly 7 days needs to be spent for recalculating the whole chain. Time spent can exponentially be increased if the chain is longer and the puzzle takes more than 10 minutes to solve. The longer the chain, the more secure it is. When these blocks are added to the chain, a copy is sent to everyone participating in the network. This is how trust is established as so many people have a copy. Let's assume100 people have copies of the whole chain and someone tries to change one of the blocks. They might add this into their chain, but there will 99 other people who do not agree with their change.

A decentralized peer-to-peer network makes it extremely hard for individual participants or groups of participants to gain control of the entire system. All participants of the network are equal and are connected to the network according to the same consensus protocol. Participants can be individuals, government bodies, organizations or associations of all types of them. The system records the chronological order of transactions within all nodes of the network that have recognized the validity of transactions through the chosen model of consensus. The result is non-reversible transactions coordinated by all network participants in a decentralized manner.

2.2 Public and private blockchains

In a public blockchain, anyone can write to the blockchain, and a public network of nodes is contributing to the creation and mining of blocks. Every node on the network contains a copy of the blockchain and can verify the full chain. Any internet user with a computer has the ability to set up as a node on the Bitcoin network which is a public blockchain, get a copy of the Bitcoin ledger, and start mining blocks.

Having so many people taking part in the network, competing to mine blocks with the proof-of-work algorithm is very wasteful though. The complex hashing puzzles are very time consuming to create and require a lot of electricity, which means miners are paid in Bitcoin for creating successful blocks that are added to the chain. The overall benefit is every transaction and generated block is completely public, and the users on each node can maintain anonymity [9].

Public blockchains offer the best level of trust and security, but this has led to some troubles with some large companies who wants to adopt blockchain techniques because they are not happy about having so much data opened and in the public for anyone to see or verify. The worry of having all this data public has led to the next level of blockchain called private blockchains. There has been a fair bit of controversy around the existence of private blockchains. Blockchain purists, bitcoin advocates, and online activists have maintained that private blockchains are not needed and they don't offer the full anonymity and openness of the public blockchain. Members of different organizations and industries, like financial services and healthcare to name a few, disagree and see the benefits to maintaining an immutable ledger that is private.

In a private blockchain, the company will write transactions, mine, and verify blocks. This means they can be much more efficient as blocks can be mined much quicker than opening it up to a huge network of mining nodes. Whilst running a private blockchain doesn't have the same decentralized security as a public blockchain, trusting the business to maintain their own chain is no worse than trusting it with your current data and transactions. With a private blockchain, the company that owns it can also decide who can read the blockchain transactions or have the ability to verify them. This means they have control over the privacy of the data that is recorded onto the blockchain. This is very important in regulated industries such as financial services and healthcare who have very strict rules they have to adhere to around how visible certain data is.

2.3 Blockchain use cases

Digital currencies

By bringing in the elements of trust and decentralization into the internet, we can open up lots of other possibilities. The first and most well-known use for blockchain is obviously digital currencies like Bitcoin.

Copyright

Another interesting use is that of protecting intellectual property [10]. Digital information is copied and redistributed with ease on the internet. This makes retaining and proving copywrite quite hard. A blockchain can be used to store digital signatures of people's work along with the timestamp in a blockchain, which means there is an immutable proof of ownership for their work. Because we have the timestamp stored next to the digital signature, we can prove when a piece of work, whether written work or an image was placed into the blockchain.

Identity management

Next, there are some great use cases for our financial services companies. Anti-money laundering and know your customer compliance practices for financial services companies have a strong case for being applied to the blockchain. Currently, financial companies have to perform labor intensive and expensive processes for each new customer. Know your customer costs could be slashed dramatically for cross company client verification, where once a person has been identified, the results along with scans of the main identifying documents are placed onto the blockchain and digitally signed [21]. This makes it easier for that person to be verified in the future as their details are easily stored in the blockchain. By having this data on the blockchain, it can be utilized by many financial companies, but they are all each contributing to the identity data, as well as verifying its integrity. Extending on from the idea of anti-money laundering and know your customer identity checking, we have more general identity management. Identity management has traditionally been a hard problem to solve on the internet. Having the ability to verify your identity is the cornerstone of banking transactions that happen online. Distributed blockchain ledgers give a much better way of proving who we are, along with the possibility to digitize personal documents. While this identifying data for a person can be digitally signed and installed onto a blockchain which can be used by multiple organizations to verify people's identity, the permanent nature of the blockchain makes this data impossible to modify and forge once it's on the blockchain, which is a massive plus point.

Property ownership management

Another interesting use case is the recording of land registry deeds or any other type of public and official documents. Using accessible public ledgers like a blockchain can be useful for any record keeping. A good example for this is property titles and deeds [22]. These types of documents can be victims of fraud, as well as expensive to administer. As properties are bought and sold, the details of their sale when transfer of ownership are permanently written into the blockchain.

Electronic voting

Finally, an exciting use case for blockchain, the one that this thesis covers, is electronic voting [10]. At the moment, in order to vote for a president or a prime minister, you have to go to a polling station, put a piece of paper in a box, and then trust a team of people to count the votes correctly. This system can be easily rigged. Using a blockchain, votes for candidates can be registered anonymously from a personal mobile device, and then many nodes on the network can all be verifying and validating the votes so nobody can tamper with the vote data. This could completely revolutionize elections.

2.4 Hashing

The process of hashing data is a very important technique using cryptography, and it forms a backbone in blockchain implementations. A cryptographic hash function is an algorithm that takes an arbitrary block of data and returns a fixed-sized string, the cryptographic hash value, such that any accidents or intentional change to that data will change the hash value. The data to be encoded is often called the message, and the hash value is also sometimes called the message digest, or simply the digest. The ideal cryptographic hash function has the following properties [23]:

- It must be easy to compute the hash value for any given message. This means that with any block of data, it should be easy to run a hashing function to calculate the hash.

- It should be infeasible to generate a message that has a given hash. This means it should be infeasible to generate some original data that will result in a predetermined hash code or digest.
- It should be infeasible to modify a message without changing the hash. This means if you change just a single bit in the data that you want to hash, then the resulting hash code is completely different.
- It should be infeasible to find two different messages with the same hash. This means you've got two different blocks of data that you want to create a hash code for. They should not both end up with the same final hash code. This is referred to as a hash collision.

Another way of thinking of a hash function is that of creating a unique fingerprint of a piece of data. Generating a hash digest of a block of data is very easy to do in most programming languages. There are various algorithms that can be used, such as MD5, SHA-1, SHA-256, and SHA-512.

A hash function is a one-way function. That means that once data is hashed, it cannot be reversed back to the original data. On the flip side to this, encryption is designed to be a two-way operation. Once some data is encrypted using a key, the operation can be reversed by decrypting the data by using the same key. When data is hashed, the hash will be the same every time the operation is performed, unless the original data changes in some way. Even if the data only changes by one bit, the resulting hash code will be completely different. This makes hashing the perfect mechanism for checking the integrity of data. This is useful for sending data across a network to another recipient. Before sending the data, the hash of the original data is calculated to get its unique fingerprints. Then the data and the hash are sent to the recipient. The recipient then recalculates the hash of the original data he has just received, and then compare it to the hash that was sent. If the hash codes are the same, then the data has been successfully received without any data loss or corruption. If the hash codes do not match, then the data received is not the same as the data originally sent, and that data shouldn't be trusted.

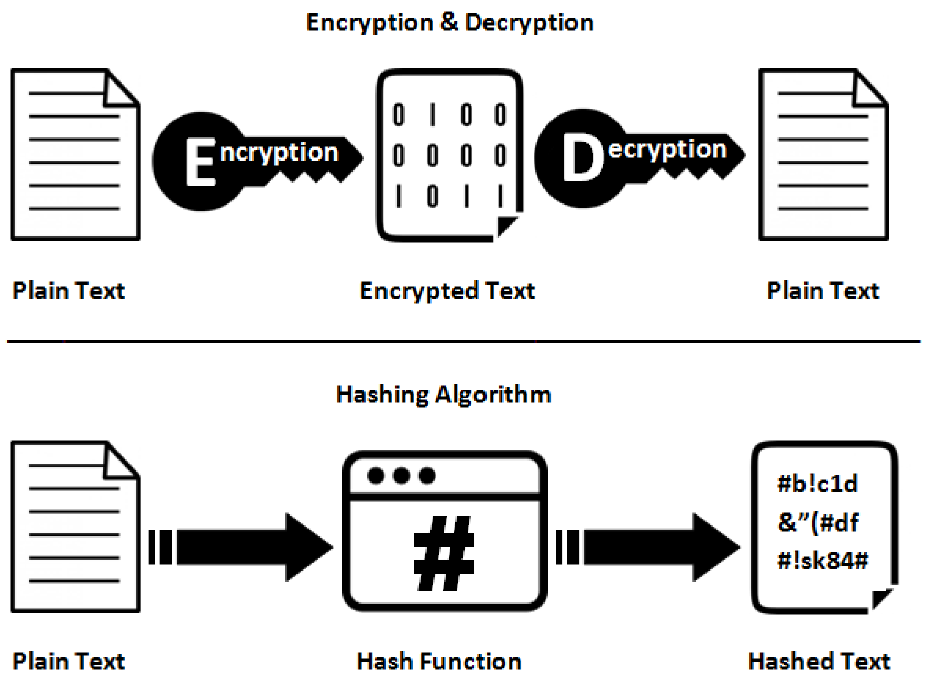

Figure 4: Difference between hashing and encryption

Source: SSL2BUY Wiki

The two most common hashing methods used are MD5 and the SHA family of hashes, SHA-1, SHA-256, and SHA-512. MD5 is generally not used much these days for new software, but it is still relevant if you are integrating with older systems that still use MD5 hashes. The SHA family is a family of cryptographic hash functions published by the National Institute of Standards and Technologies, or NIST for short. The SHA family covers some different variants including SHA-1, which is a 160-bit hash function which resembles the early MD5 algorithm. This was designed by the National Security Agency to be part of the digital signature algorithm [25]. Cryptographic weaknesses were discovered in SHA-1, and this standard was no longer approved for most cryptographic uses [26]. Next, there is SHA-2, which is a family of 2 similar hash functions with different block sizes known as SHA-256 and SHA-512. They differ in word sizes. SHA-256 uses 32-bit words, whereas SHA-512 uses 64-bit words. These versions of the SHA algorithm were also designed by the National Security Agency [27]. Finally, there is SHA-3, which is a hash function formally called Keccak chosen in 2012 after a public competition amongst non-National

Security Agency designers. It supports the same hash lengths as SHA-2, and its internal structure differs significantly from the rest of the SHA family.

2.5 Block structure

The block is one of the most essential and fundamental parts of the blockchain ledger data structure. Blockchain blocks where a block contains a single transaction consist of two parts: a transaction details and header. Every field related to the actual business logic is stored in the transaction details part. A series of fields that do not contain any transactional details but are needed to link the blocks together and to make them cryptographically secure are stored in the header part.

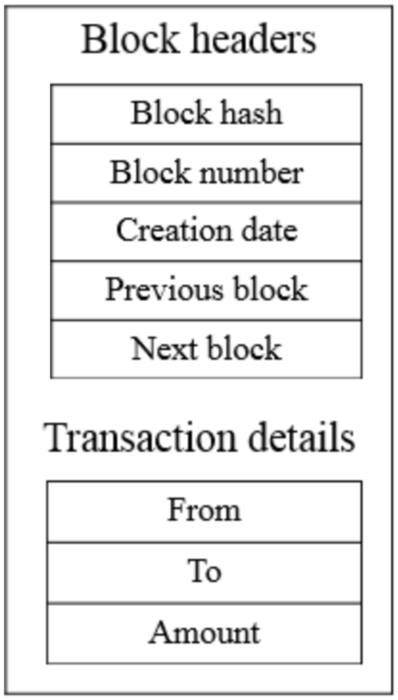

Figure 5: A block structure

Block header fields

Block hash – Block hash is a unique fingerprint for all of the data that's going to be contained in the block.

Block number – Block number is the number of the block in the chain. As new blocks are written into the chain, the number increments. The first block in the chain is referred to as the genesis block.

Creation date – Creation date is a date stamp for when the block has been created.

Previous block – Previous block is a reference to the hash of the previous block. I'll talk about this in more detail in a moment.

Next block – Next block is a reference to the next block in the chain

A block has the transaction details, a series of business logic related fields, along with the block number, the creation date, and the previous block hash. These concatenated fields are run through the hashing function to create the unique hash code. The results of which is then stored in the block hash field. The hash function is a one-way operation that is calculated and becomes a unique fingerprint for the data that was passed into the hashing function. If just a single binary bit of any of the originating information in the transaction is modified, then the hash will be completely different, as this property of a hashing function is important in detecting any changes to the data in a block chain.

Once the hash within the single block is calculated, the blocks can be linked together easily. In the block header, there is a field called the previous block hash and the next block. It points to the next block in the chain. The previous block hash contains a block hash from its parents.

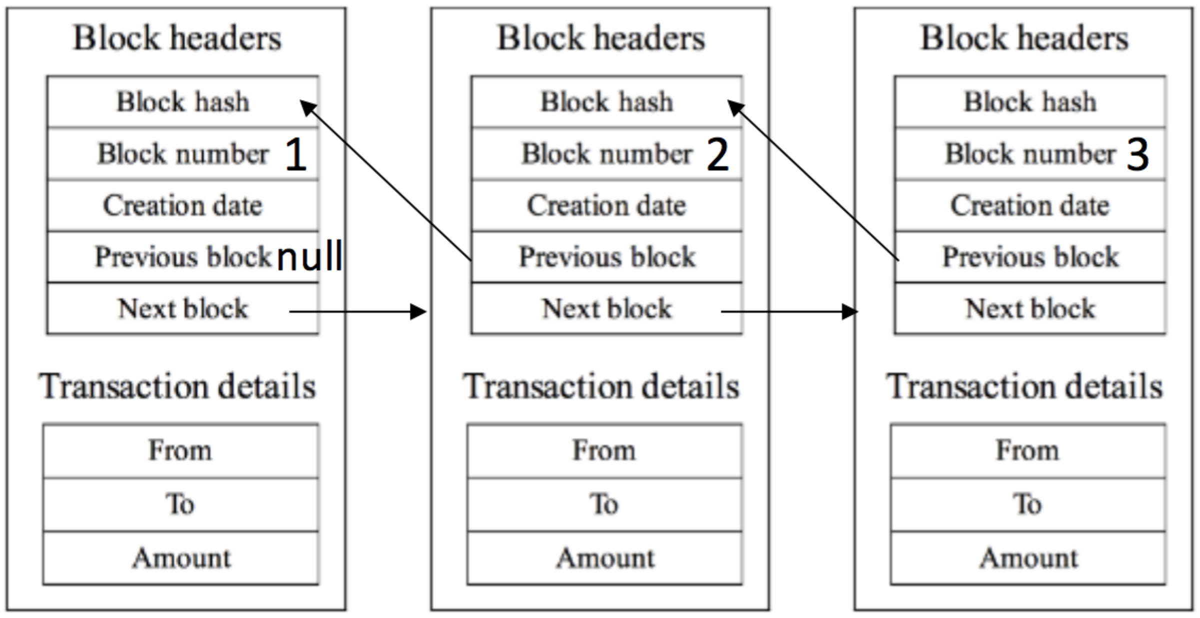

Figure 6: Linked blocks. Note: The first block has no parent. Therefore, the previous block value is null.

As visualized in the figure 6, in the linked blocks, first there is a block number 1. The block hash is calculated with this block and the result is stored in the block hash field. Because this is the first block in the chain, the previous block hash field will be empty, as there are no previous blocks. Then the second block is created and the block hash of the first block is passed in. This block hash is stored in the previous block hash. Then the block hash for block 2 needs to be calculated as previously. By including the previous block hash into that second block's hash code, we are cryptographically linking those two blocks together. This characteristic is important for the chain verification. Then in order to add the third block into the chain, first of all, the block is created and the required transaction details are passed in. Then the block hash of the current head of the chain is passed and it is stored in the previous block hash. Later the block hash for the new block is calculated, which creates a cryptographic link between the two blocks as before.

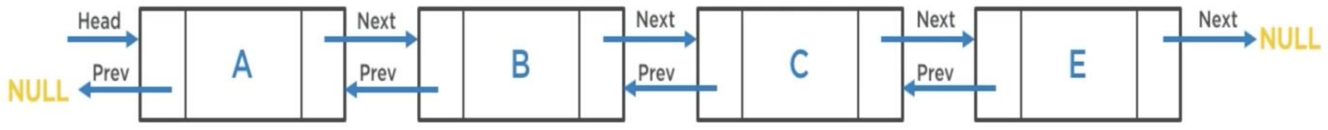

Figure 7: Doubly linked list

In a block chain, the first block, or genesis block, is the only block that will have an empty previous block hash. Conceptually this is the same as a double linked list as shown in the figure 7. Each node in the list has two fields that need to be provided: the next node and the node before it. It is the same as the linked blocks, except cryptographic hashes are used to represent the link back to the previous block. Once a chain of blocks containing the block header and the transaction details is built, performing a verification process is necessary in the blockchain to ensure that all the blocks have maintained their integrity, and that no data has changed after the block has been inserted into the blockchain. If, for example, there were 5 blocks in the chain, and some days that are in a transaction has been changed at block 2, during the verification of the chain, failure of the block 2, 3, 4 and 5 are expected. This is because a block hash from a block is also included in the block hash of the next block. If that block's data changes in any way, then that block hash will change, which means the block hash of the next block will also be difference, because of the inclusion of the previous block hash. This is one of the strongest features of the blockchain, because cryptographic hashes and proofs are used to check the integrity of the chain. Changing the integrity of a single block will lead the verification failure of the rest of the chain that follows after that block. This makes tampering very easy to spot.

2.6 Blocks with multiple transactions, Merkle tree

It is more common to have multiple transactions contained within a block. In the multiple transactions block, the mechanics of how the block operates with the block hash being used to link the blocks together is the same as in a single transaction block. In such blocks, a single hash code that represents all of the transactions stored in the block is generated by using a data structure is called a Merkle tree.

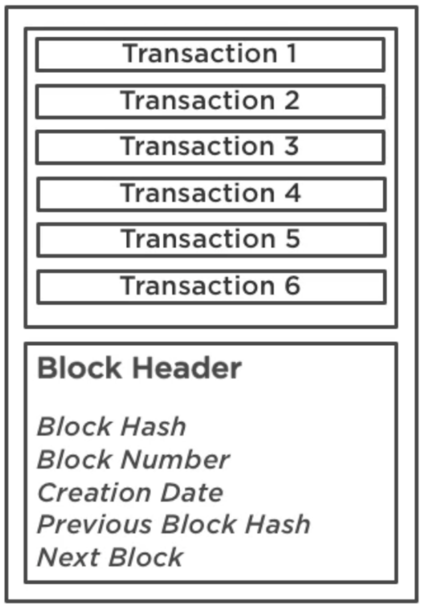

Figure 8: A block with multiple transactions

In the figure 8, four transactions are described. In those nodes a hash is created using our SHA-256 hash function. Then the hashes or transactions 1 and 2 are taken and they are hashed together to produce a new hash, which is called hash (1,2). This box represents a single hash of the 2 hashes from transactions 1 and 2. Then the same is applied for transactions 3 and 4, which produces a new hash code called hash (3,4). Once this layer is created in the tree, then create a new node at the top is created, which is a hash of the combined hashes for hash (1,2) and hash (3,4). The example in this figure is three layers deep. As there are more transaction nodes at the bottom of the tree, more layers will be introduced into the Merkle tree, because each of those individual transactions are hashed in pairs.

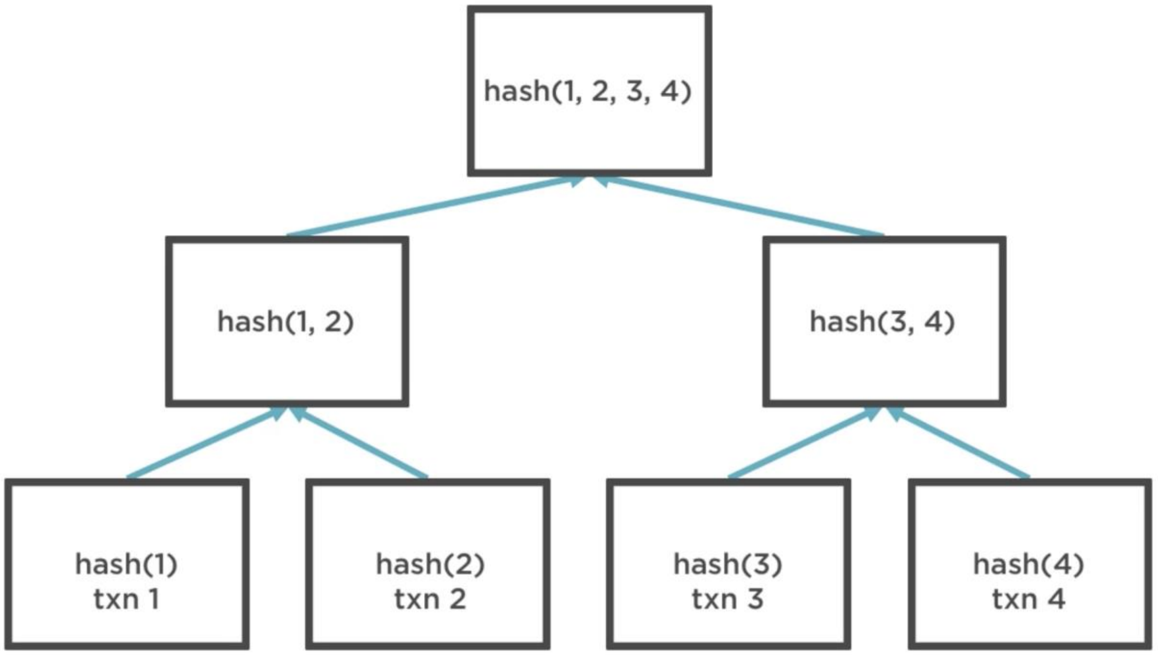

Figure 9: A Merkle tree that represents four different transactions.

Merkle tree is a structure that allows for efficient and secure verification of content in a large body of data. This structure helps verify the consistency and content of the data. It takes a number of pieces of data and then calculate a cascading tree of hashes of that data, which results in a single hash that represents all of the different pieces of data. Incorporating a Merkle tree, instead of a single hash representing one transaction, the hash at the top of the tree is used to represent the multiple transactions. In the figure 9, the root Markle tree hash is called hash(1,2,3,4), and it represents all of the transactions.

2.7 Consensus, The Byzantine Generals' Problem, Proof of work

The Byzantine Generals' Problem was first described by Leslie Lamport, Robert Shostak, and Marshall Pease in a 1982 paper, The Byzantine Generals' Problem. The problem states that there is a group of generals who each command a unit in the Byzantine army.

Let's assume there are two armies on opposite hills and a city in a valley. The armies both want to launch an attack on the city, but there is a problem. To do this successfully, they will have to attack at the same time; otherwise, the city's defenses could overpower a single army in the attack. To coordinate the attack, the generals have to send messages to each other on each hill. The first general from army A will send a message to army B saying we will attack at 5 a.m. on Tuesday. Then, army B will send a reply to army A, confirming that they have received the message. This sounds fine in theory, but what happens if guards in the city capture and imprison the messenger, and then send a spy who delivers a different message, which is attack at noon on Friday, there is no need to send a reply. This is nearly four days later.

It has been said that this problem has no real true solution, even when we factor in computers and modern communications for stopping message intercepting. Computers can communicate reliably, but there is never 100% degree of certainty that computers communicate reliably. It is possible to make the communications very reliable, but never totally foolproof. In the example above with the armies, it was intended for the recipient of the first message, army B, to send a response to confirm the original attack days. But what happens if the messenger with that reply gets caught, and the message is modified? You could request an additional acknowledgement response, but that messenger could get caught, so you end up in this loop of waiting for responses. There are many practical solutions to this problem which are good enough for real-world communications. When Satoshi Nakamoto first described bitcoin, he proposed a method that makes it very hard for the city to scupper the plans of the generals and their armies by changing their message [6]. The message from the generals of the other armies has the same bearing as additional currency or a transaction on a blockchain. Instead of a message like, attack at 6 a.m. on Tuesday, the message could be, "Pay Bob 100 dollars". The key feature here is that once a message has been sent or a transaction has been put onto the blockchain, it can't then be changed after the fact.

Solving problem with hashing

First, the generals will take their message of "Attack at 6 a.m. on Tuesday", and will append some additional to the end of the message. This piece of extra data is called a nonce. So now the message looks like "Attack at 6 a.m. Tuesday 3FrgH542dFe". The generals then agree to set a message policy where the hash of that complete message must result in a hash that starts with six 0s.

Hash("Attack at 6 a.m. Tuesday 3FrgH542dFe") = 000000347912509347912509

If the hash doesn't start with that number of 0s, then the message is fake. The nonce added to the end of the message is what determines how many 0s a hash starts with. In the example above, a nonce is 3FrgB542dFe. That nonce can be treated as a number. They didn't just guess that nonce, they had to calculate it. They start a nonce at 0, and they keep adding 1 to it until the resulting hash has the desired number of 0s at the front of it. This means in the example above the final hash looks like what you can see on the screen now, so we hashed a message, attack at 6 a.m. Tuesday without a long nonce, and the resulting hash code starts with six 0s. This takes time to calculate; it's not an instant operation. When they first start out, the nonce would have been 0, so the hash operation might rule out, by instead of starting with six 0s, it starts with 384723. So they add 1 to the nonce and try again. They keep doing this until they reach a desired goal. To find the correct nonce for the hash requires a lot of computing power, as the nonce should be incremented by 1 each time until the solution is found to the hash puzzle, which in this case is six 0s at the beginning of the hash. The problem is significantly easier to recalculate, if we need to find three 0s. Alternatively, the problem is considerably more complicated to calculate, if there needs to be ten 0s.

In the example of the generals, generals write the message and then they try to get their communication experts start calculating the hash. Let's assume this takes six hours to figure out, because they have to try billions of combinations. Once they have found the correct nonce, they send the message over to the other generals and their waiting armies. To validate that the message is correct, the general merely calculates the message again with the given nonce and checks the number of 0s. If it is six then the message is genuine, if it is not six then the message is not valid. If the message were intercepted and changed, the person changing a message would have to go through the same process of calculating the nonce to get the hash that starts with six 0s. Once the second general has received a message, if they wish to send a reply back to the other general, they have to go through exactly the same process of finding a correct nonce to give the required number of 0s at the start of the hash.

The nonce is time-consuming to calculate for the message sender, but a message is straightforward to verify the recipients. This technique is known as proof of work. This means the message sender proves their authenticity by spending a lot of effort doing the work to calculate the nonce to solve the hashing puzzle. Let's assume that the city is now familiar with the technique being used to hash a message. In order to protect themselves, they buy a massive super computer. They then catch a messenger and have the computing power to change the message and recalculate the nonce, but faster than the generals of the armies. It may take the generals 10 hours to calculate the correct hash, but the town can now do it in 1 hour with this powerful super computer. Now the general of the other army is getting conflicting messages, but when they verify the hashes, they all work out correctly.

Let's assume, instead of only two generals in their armies, there are many generals and more cities. The overall goal is the same for the generals, as they want to get their message to the other generals without fear of them being modified. The cities want to scupper these plans just as they did before. This is the fundamental idea Bitcoin tries to solve. The generals all combine their messages into a single block. In this new scheme, the generals increase the complexity – the number of 0s to 10. This means it takes a significant amount longer to calculate the hash. Each army is from a different country, and they each have their own various computing resources available to them. Each of the armies received all of the messages for the block, and they each race to calculate the hash and find the nonce that allows them to solve the hash puzzle. In cryptocurrency terminology, this race to solve the hash puzzle is called mining. It only takes one army to find the correct nonce. Once found it, they share it with their allies in the other army straightaway.

From the perspective of the cities, the idea of outperforming the combined efforts of the armies seems impossible, as there is so much computing power looking for the mined hashes. This could be possible if only a few armies were competing to find the block hash, and the city has purchased significant computing power and may have teamed up. This in Bitcoin terms is referred to as the 51% attack. It works when there are different cities that have the same amount of computing power or more as the armies combined. But if there are 20 armies, this starts becoming feasible. This is why many cryptocurrencies use public blockchains. The more nodes try to calculate the block hash, the more secure it is. If 5000 nodes of miners working to calculate and validate blocks, it is not impossible to defeat the system, but it has been significantly harder with strength in numbers and computing power. This proof of work technique does have its problems, though, and that by its very nature of solving this complex hashing puzzle, it consumes a considerable amount of both computing power and electricity to power these computers.

2.8 Consensus maintenance. Longest chain

Each node in a blockchain network works in two states. The node is either verifying blocks that were created by the peers or working to solve the proof of work hashing puzzle to create a new block for a series of transactions. Each node is independent of each other, so they are not synchronized in any way. They each react to their inputs accordingly. Due to network latencies of blocks being sent around and transactions being sent to the nodes, each node will not necessarily have identical information to work with. Messages could get lost in transit or arrive in different orders. Because of this, the two modes that a blockchain node operates in will be in different states per node. Because of this, maintaining a consistent blockchain transaction history between nodes is quite challenging. This means the challenge is to get one unambiguous history of transaction data, even though there are message delivery delays and reliability issues between nodes. The nodes should be kept independent of each other without resorting to a centralized solution. As the blockchain nodes are entirely independent, they are therefore responsible for coming to an agreement of which version of the transaction history is selected by themselves. The premise for how they choose transaction history is based on how new blocks are picked, added to the block and protected from manipulation is in the proof of work algorithm. The proof of work algorithmic process makes a block creation process computationally very expensive, and efforts to manipulate or change transactions in a blockchain even more computationally expensive. This means that the amount of total computational efforts spent creating blocks is good criteria for selecting the transaction history. If all of the participating nodes follow the same selection criteria for the transaction history, nodes will all eventually agree on an identical version of the transaction history. By using the idea of the amount of computational effort to select the history, this leads us to a strategy to pick the correct transaction history. This method is referred to as picking the longest chain.

Longest chain

The longest chain solution revolves around the idea that a chain or fork in a blockchain that consists of the most blocks will represent the most combined computational efforts with calculating the proof of work hashing puzzle for the block.

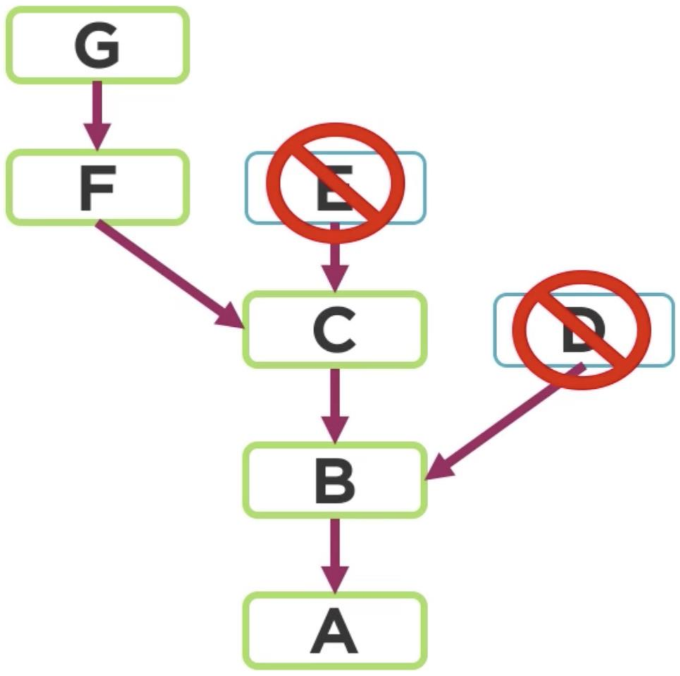

Figure 10: Blockchain with forks.

First, let's assume a case where all of the nodes on a network have an initial blockchain agree that consist of two blocks. These are represented by the boxes A and B as shown in the figure 10. Each box is a block in the blockchain. The makeup of a block is quite complicated, as we have multiple transactions and a block header. The letters in a block boxes represent the block hashes. As new transactions arrive at each node in the network, they each race to solve the proof of work hashing puzzle. The chain after a node has won the race to create the next block – the block C, and sends that block to all of the other nodes. This is the best-case scenario where the race to solve the hashing puzzle is won in all of the nodes except the new block. In this case, all of the nodes have a consensus on the blockchain. If the winning block is delayed getting through to all the other nodes, another node finds a winning block hash, which then starts to distribute to the other nodes. Once a block is finished propagating, it would create a fork in the blockchain where both blocks C and D are linked to block B. In fact, this is more like a block tree than a blockchain. In that scenario, a full consensus cannot be reached as the notion of picking the longest chain is impossible, because both branches have the same length. A node can decide for themselves which branch to extend. They could try to build the next block on top of block C, or they could do it on top of block D. At this stage, there is no wrong answer, both are valid. So the nodes go off and all try to create new blocks. At around the same time, two nodes of the hashing puzzle will send out their winning blocks. In this case, both of the nodes have built their block on top of block C. In this case, the chain that is A, B, D can be rolled out, as this is the shortest chain in the tree. The loads again all go off and race for the construction of the next block. The nodes can either construct the next block so that it builds off block E, or off block F. Eventually one of the nodes solves the hashing puzzle and sends a block to all of the nodes on the blockchain network. This winning node, G, was built on top of node F. The chain A, B, C, F, and G is five blocks long, whilst A, B, C, E is only four blocks long. The branch that is five blocks long wins. Consensus is reached as every node is performing this same role.

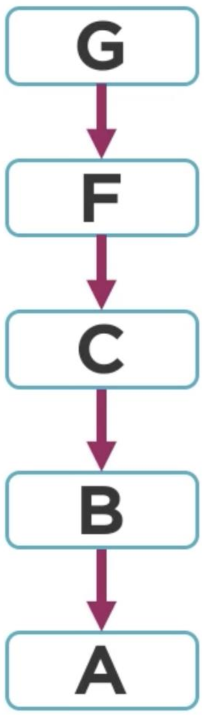

Figure 11: Final blockchain after the consensus is reached.

Applying the longest chain process to the branches of the tree to set the common authoritative chain between the nodes does have some side effects. These are orphaned blocks and reclaiming mining rewards. When there are multiple nodes competing to mine blocks of transactions by solving the proof of work puzzle, there are usually forks in a blockchain. This means the blockchain structure is more like a block tree. As the nodes perform the longest node selection criteria over the tree, we end up deciding as to the authoritative branch to use. This means that the blocks that lose out in the selection are orphaned from the blockchain. In the figure 11, the blocks D and E are orphans. This means the transactions contained within those blocks no longer have a home in the blockchain. Now that these transactions have lost their chance of being part of the blockchain, they need to be given another chance. Then those transactions are placed back into the port of transactions for that node, and they are given the opportunity to be mined again as part of another block. This means that transactions that once appeared as though they were added to the blockchain suddenly seem to disappear from existing for a while until they are re-added into the blockchain as part of a new block.

3 Digital signatures

3.1 Overview

An important function of cryptography is to ensure non-repudiation of a sent message. A valid digital signature gives the recipient a reason to believe that the message was created by a known sender, such that the sender cannot deny having sent that message. Digital signatures give you both authentication and non-repudiation. Authentication because the signatures had to be created by a user with a valid, private key, and non-repudiation as a receiver can trust that the message was signed by a known sender, as only they know the private key [11].

For the receiver of the message, a digital signature allows the receiver to believe that the message was sent by the correct sender. This can be thought of as the digital equivalent to a signature on a letter, except a digital signature is much harder to forge. A digital signature consists of the following three algorithms:

1. the public and private key generation using RSA,
2. a signing algorithm that uses the private key to create the signature,
3. a signature verification algorithm that uses the public key to test if the message is authentic.

The private and public keys are two keys that are mathematically linked. The public key, as the name suggests, can be known by anyone, and the private key should only be known by the sender of the message. So therefore, the private key that has to be kept secret.

In the following example, Alice is going to send a message to Bob, and that message will be signed with a digital signature.

Alice	Bob
1. Encrypts the data 2. Takes a hash of the encrypted data 3. Signs the data with her private key 4. Sends the encrypted data, the hash, and the public key.	1. Calculates the hash of the encrypted data 2. Verifies the digital signature using the public key

First, Alice encrypts some data that she wants to send to Bob. For this example, it doesn't matter whether the data was encrypted with a symmetric or asymmetric encryption algorithm. Once this data has been encrypted, Alice takes a hash of that data. Next, Alice signs the data with her private signing key. This creates a digital signature. Then, Alice sends the encrypted data, its hash, and the signature to Bob. First, Bob recalculates the hash of the encrypted data. Bob then verifies the digital signature using the calculated hash and the public signing key. This will tell Bob whether the signature is valid or not. If it is valid, Bob can be confident and see it was Alice that sent him the message in the first place, as it could only have been signed using her private signing key, which only Alice knows. If this signature was not valid, then Bob should not trust the origin and authenticity of the message. As mentioned, the digital signature signing is based on RSA. With RSA, the data can be encrypted with the recipient's public key, and then the recipient decrypts it with their private key. With a digital signature, signing and verifying is the other way around. When the sender signs a message, they use their own private key, and then the recipient verifies the signature using the sender's public key. It is due to the fact that we've signed the sender's private key that a recipient can trust that a message was sent by that sender, as it will only be them that knows that private key.

3.2 Ring signatures

A ring signature is an electronic signature that allows one of the members of the group (called the ring) to sign a message on behalf of the whole group, and it will not be known exactly which of the group members made the signing. The ring signature algorithm was created by R. Rivest, Adi Shamir and Ya. Tauman and was announced in 2001 at the international conference

Asiacrypt [12]. The authors emphasize in the paper the absence of a central or coordinating structure in the formation of this signature: "The ring are geometric regions with uniform periphery and no center."

The ring signatures have the following properties:

- Unconditional Anonymity: The attacker cannot determine which member of the ring the signature was generated from, even if the ring member's private key is obtained, the probability does not exceed 1/n.
- Correctness: The signature must be verifiable by all others.
- Unforgeability: The other members of the ring cannot fake the signature of the real signer. The external attacker cannot forge a signature for the message m even if it is based on a valid ring signature.

3.3 One-time ring signature

One-time ring signatures allows a user to sign multiple messages in a way that the verifier can find out that the messages are signed by the same user without revealing the user's actual identity. This solution is introduced by Nicolas van Saberhagen in the paper CryptoNote [13] lies in using a different signature type than those currently used in electronic cash systems.

The one-time ring signatures described in the paper uses the fast scheme EdDSA, introduced by D.J. Bernstein in the paper "High-speed high-security signatures" in 2012 [14]. EdDSA is based on the elliptic curve discrete logarithm problem [15].

One-time ring signature has the following parameters:

$\boldsymbol{q}$: a prime number; $\boldsymbol{q = 2^{255} - 19}$;

$\boldsymbol{d}$: an element of $\boldsymbol{F_q}$; $\boldsymbol{d = -121665/121666}$;

E: an elliptic curve equation; $-x^2 + y^2 = 1 + dx^2y^2$;

G: a base point; $G = (x, -4/5)$;

l: a prime order of the base point; $l = 2^{252} + 27742317777372353535851937790883648493$;

HH_s: a cryptographic hash function $\{0, 1\}^* \rightarrow F_q$;

H_p: a deterministic hash function $E(F_q) \rightarrow E(F_q)$.

A one-time ring signature contains four algorithms: **(GEN, SIG, VER, LNK)**:

GEN: takes public parameters and outputs an ec-pair (P, x) and a public key I.

SIG: takes a message m, a set S' of public keys $\{P_i\}_{i \neq s}$, a pair (P_s, x_s) and outputs a signature σ and a set $S = S' \cup \{P_s\}$.

VER: takes a message m, a set S, a signature σ and outputs “true” or “false”.

LNK: takes a set $I_s = \{I_i\}$, a signature σ and outputs “linked” or “indep”.

The idea behind the protocol is fairly simple: a user produces a signature which can be checked by a set of public keys rather than a unique public key. The identity of the signer is indistinguishable from the other users whose public keys are in the set until the owner produces a second signature using the same keypair.

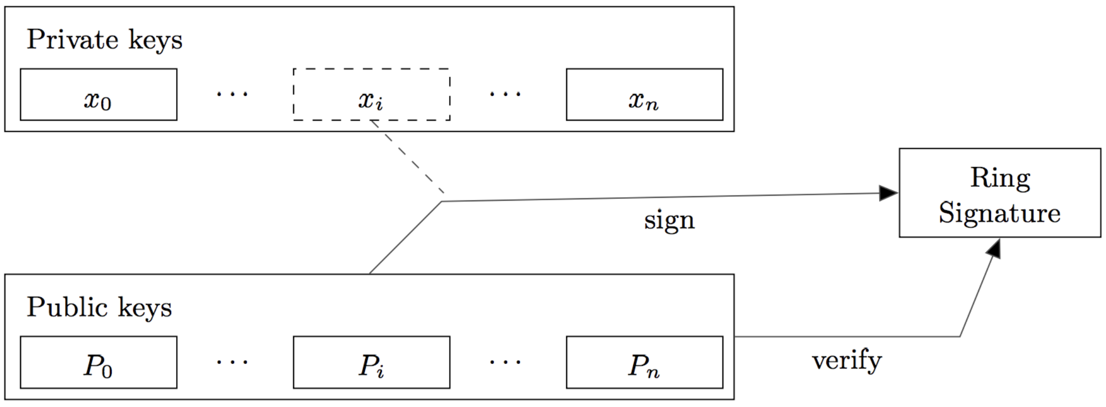

Figure 12: Ring signature anonymity

GEN: The signer picks a random secret key $x \in [1, l - 1]$ and computes the corresponding public key $P = xG$. Additionally, he computes another public key $I = xH_p(P)$ which is called the "key image".

SIG: The signer generates a one-time ring signature with a non-interactive zero-knowledge proof using the techniques from [21]. He selects a random subset S' of n from the other users' public keys P_i , his own keypair (x, P) and key image I. Let $0 \leq s \leq n$ be signer's secret index in S (so that his public key is P_s). He picks a random $\{q_i \mid i = 0 . . . n\}$ and $\{w_i \mid i = 0 . . . n, i \neq s\}$ from $(1 . . . l)$ and applies the following *transformations*:

$$L_i = \begin{cases} q_i G, & \text{if } i = s \\ q_i G + w_i P_i, & \text{if } i \neq s \end{cases}$$

$$R_i = \begin{cases} q_i \mathcal{H}_p(P_i), & \text{if } i = s \\ q_i \mathcal{H}_p(P_i) + w_i I, & \text{if } i \neq s \end{cases}$$

The next step is getting the non-interactive *challenge*:

$$c = \mathcal{H}_s(m, L_1, \ldots, L_n, R_1, \ldots, R_n)$$

Finally, the signer computes the *response*:

$$c_i = \begin{cases} w_i, & \text{if } i \neq s \\ c - \sum_{i=0}^{n} c_i \mod l, & \text{if } i = s \end{cases}$$

$$r_i = \begin{cases} q_i, & \text{if } i \neq s \\ q_s - c_s x \mod l, & \text{if } i = s \end{cases}$$

The resulting signature is σ.

$$\sigma = (I, c_1, \ldots, c_n, r_1, \ldots, r_n)$$

VER: The verifier checks the signature by applying the inverse transformations:

$$\begin{cases} L_i' = r_i G + c_i P_i \\ R_i' = r_i \mathcal{H}_p(P_i) + c_i I \end{cases}$$

Finally, the verifier checks if the following equality is correct, the verifier runs the algorithm **LNK**. Otherwise the verifier rejects the signature.

$$\sum_{i=0}^{n} c_i \stackrel{?}{=} \mathcal{H}_s(m, L_0', \ldots, L_n', R_0', \ldots, R_n') \mod l$$

LNK: The verifier checks if *I* has been used in past signatures (these values are stored in the set I_s). Multiple uses imply that two signatures were produced under the same secret key. The meaning of the protocol: by applying *L*-transformations the signer proves that he knows such *x* that at least one $P_i = xG$. To make this proof non-repeatable we introduce the key image as $I = xH_p(P)$. The signer uses the same coefficients (r_i, c_i) to prove almost the same statement: he knows such *x* that at least one $H_p(P_i) = I \cdot x^{-1}$.

If the mapping $x \rightarrow I$ is an injection:

1. Nobody can recover the public key from the key image and identify the signer;

2. The signer cannot make two signatures with different *I*'s and the same *x*.

4 TOR anonymity network

4.1 Overview

TOR (The Onion Router or Multilayer Router) uses a network of private computers called nodes that forward and encrypt Internet traffic. TOR initially started to be developed with the support of the US government by the Office of Naval Research and DARPA [16].

Usually on the Internet, the data is transferred from the source to the destination over a global network of public servers. In its structure, the World Wide Web is an open, free space that allows data to move freely from the source to the destination. Nevertheless, this design has a number of drawbacks in terms of security.

For comparison, the TOR network from the first step passes traffic exclusively through its nodes – input, transmission and output, which each has its own cipher. Due to this configuration, none of the nodes in the TOR network have information about the source of the request and its destination, or even network ciphers, which protects information about users.

Anonymous transfer of data from one node to another in the subnetwork TOR has established itself as a reliable protection tool, it is very difficult to crack.

5 CryptoVote

5.1 Overview

We are going to introduce a new election protocol called – CryptoVote. In order to make the election procedure more decentralized, transparent and secure, the protocol uses blockchain and one-time ring signatures described in the paper CryptoNote 2.0 by Nicolas van Saberhagen [13] which itself is using EdDSA [14] as a base signature. We are using the same parameters and algorithms for the ring signatures as described in the section 3.3 of this thesis. Within the scope of this protocol, we are going to solve the following problems mentioned in the requirements below.

5.2 Protocol requirements

Eligibility – Only the eligible voters should have a right to vote.

Single-vote – An eligible voter should have a right to vote only and only once.

Anonymity – Identity of the voter should be anonymous. No one but the voter himself is able to verify to whom he has voted.

Voter verifiability – The vote should be verifiable without being able to reveal the identity of the voter.

Fairness – Partial results should not be able to be counted by the public before the voting is finished in order not to have influence over the voter.

Integrity and correctness – Once the vote is recorded, reverting back the vote or changing the vote should not be possible. No one should be able to have any influence on the infrastructure during the voting process for preventing them to violate the integrity.

Universal Verifiability – Everyone should be able to verify the results of the elections.

Availability – Election system should be fault tolerant.

5.3 Related works

There have been several proposals [17,18,19,20] on blockchain based election protocols. However, we haven't found a solution to meet all the requirements listed above.

The protocol provided by Yi Liu and Qi Wang [17] is the closest to meet the requirements, but their solution still does not solve the requirement fairness. Within their solution, the partial results can be announced, and it might influence voter's decision during the voting phase. Additionally, their solution is a target to the 51% attack [see chapter 2.7].

Some other solutions use either uses a crypto currency network as its underlying network which causes issues within the eligibility of the voters or they are target to the 51% attack. Besides from that, they also don't meet the fairness requirement [18, 19, 20].

5.4 Definitions

Election commission – a central authority who handles a voter registration and provides the public and the private key pair of the election. The public/private key pair is used to encrypt and decrypt the voter’s vote.

Voter – a person who is eligible to vote in the elections.

Observer – a blockchain network node that can be added by any individual, which runs and validates the votes according to the consensus.

The public key of the election, P_e – a standard elliptic curve public key that is provided by the election commission before the voting in order to encrypt the voter’s vote.

The private key of the election, S_e – The corresponding private key of the standard elliptic curve key pair of the election that is provided by the election commission after the voting in order to decrypt the votes for counting.

Voter's public key, P_v – a standard elliptic curve public key which is linked to the voter's actual identity. It is derived from the voter's private key.

Voter's private key, S_v – a standard elliptic curve private key which the voter uses to sign her vote. It is derived from the voter's secret.

Voter's key image, I_v – a key which represents the voter in the blockchain. It is derived from both the public key and the secret (see: **GEN** function in the section 3.3 of the thesis). It is practically impossible to link the key image to the corresponding public key. **Voter's signature, Σ_v** – a one-time ring signature used to validate the voter's eligibility.

5.5 System design and components

In order to improve transparency, we are using the blockchain network to store the votes within the observer nodes of the network. Considering the blockchain will be open to everybody to read any time, we need to find a way to prevent revealing the voter's identity, at the same time we should be able to validate the authenticity of the vote. We can solve this problem by using the one-time ring signatures (see: section 3.3). Using the ring signatures, a verifier is convinced that the real signer of the message is a member of the group, but he cannot exclusively identify the signer.

5.5.1 Election commission

Although the system is decentralized during the actual voting and the vote verification process, in order to keep the list of the eligible voters, it needs to have some centralization. Therefore, we need to have a registrar, which is a government authority that registers the voters in the system, then provides the list of the eligible voters before the voting phase. The election commission owns a database which maps the voters' identity and their public key in their system. Additionally, it provides voters a single public/private key pair for the election to encrypt and decrypt the votes.

Pre-voting phase

Before the voting phase, the election commission accepts the public keys from the voters, confirms the voter's eligibility by checking their ID cards, and add the public key to their database. After the registration is done, the commission publishes the mapping of the list of the eligible voters and their public keys online. Additionally, the election commission announces $\boldsymbol{P_e}$, the public key of the election to voters, so that they can encrypt their votes using it.

Voting phase

During the voting phase, an election commission needs to run at least one observer node.

Post-voting phase

After the voting is finished, the election commission announces $\boldsymbol{S_e}$, the private key of the election, so that everyone can use it to decrypt and count the votes. During the vote decryption, the voter's actual identity is not revealed as the voter uses ring signature to hide his identity.

5.5.2 Voter

Pre-voting phase

Before the voting, the voter needs to generate his private key – $\boldsymbol{S_v}$, the public key – $\boldsymbol{P_v}$, and the key image $\boldsymbol{I_v}$. The voter registers his public key within the election commission registrar by providing his ID card. Remembers his private and public keys for the election.

Voting phase

During the voting, first, the voter needs to encrypt his vote using the public key of the election. We need to make it very hard or practically impossible to anyone without the private key of the election to find out to whom the vote is given, in order to meet the requirement fairness. Considering there will be a limited number of the candidates in the elections, encrypting the candidate name would not be efficient. Because the ciphertext for all the candidate names can easily be generated within the public key. We need to make sure that the ciphertext of the vote is unique in each vote entry. Therefore, we are going to represent the vote by concatenating the voter's key image, the candidate name and randomly chosen number between $N_r \in [1, 10^9]$ using a delimiter. So finally, we will encrypt the string in the following format:

$$\boldsymbol{V = I_v|CandidateName|N_r}$$

$$\boldsymbol{V_{cipher} = encrypt(P_e, V)}$$

Then the voter picks 10 random voters from the eligible voter list and creates a one-time ring signature using the encrypted vote, the random eligible voters' public keys, his public key and the private key.

Let's assume, a set of the voter and the random eligible voters' public keys – $\boldsymbol{L \subseteq \{P_{v1}, P_{v2}, P_{v3}, \ldots, P_{v10}\}}$

$$\boldsymbol{\Sigma_v = SIG(V_{cipher}, L, \{S_v, P_v\})}$$

Finally, the voter creates a new blockchain transaction using the ciphertext of the vote, the public keys of the ring signature members and the ring signature, and broadcasts it to the network.

Note: Based on the chapter 3.3, the ring signature itself holds the key image. Considering the resulting $\boldsymbol{\Sigma_v}$ is holding $\boldsymbol{I_v}$ which represents the voter without revealing his real identity, storing just the ring signature $\boldsymbol{\Sigma_v}$ in the block is enough for complying the single-vote requirement.

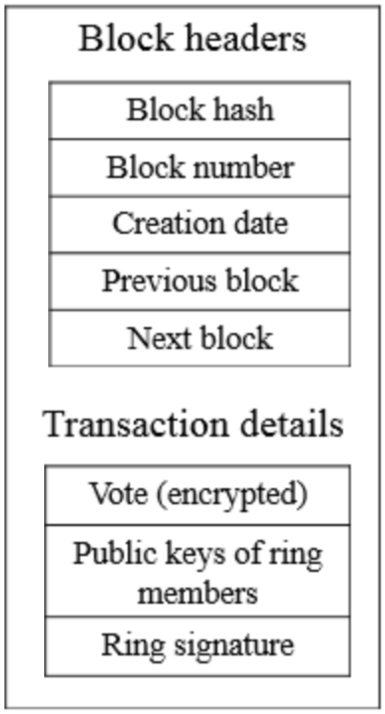

Figure13: CryptoVote block structure

Post-voting phase

No involvement is required from the voter in the post-voting phase.

5.5.3 Observer

The main purpose of an observer node is to run the infrastructure and validate the votes according to the consensus. An observer node can be added by any individual to the blockchain network.

Pre-voting phase

Before starting observing, the observer node needs to get the list of the eligible voter public keys from the election commission. An observer can ask another observer to receive his last available chain.

Voting phase

During the voting phase, observer listens to the new incoming transactions and the chain requests. When the observer receives a new transaction, it validates the transaction using the transaction validation rules below:

1. Every ring member in the ring members field of the transaction is an eligible voter according to the list published by the election commission.
2. There is not any other transaction within the same key image $\boldsymbol{I_v}$ in the chain.
3. The ring signature is valid: $\boldsymbol{Valid = VER(V_{cipher}, \Sigma_v, L)}$

If the transaction is valid, the observer creates a new block and add it to the chain, otherwise, it rejects the transaction.

When the observer receives a chain request, it sends back its own copy of the blockchain.

Post-voting phase

After the election is finished, observer nodes need to find out the most correct chain by making the chain request to other observers and synchronize their own copy of the chain with the most correct one. The observers agree on the most correct chain using the following CryptoVote chain validation rules:

1. The new chain length needs to be larger than the local chain length.
2. The chain does not hold a pair of transactions where the key image is the same.
3. The chain should be valid according to the blockchain validation rules: the previous block hash field in the given block should be equal to the actual hash of the previous block.
4. Each transaction within the chain should be valid according to the transaction validation rules (see: section 5.5.3 voting phase).

By applying these rules to each chain received by another observer, the observers agree on the most correct chain.

Once the most correct chain is found, the observer nodes still need to be running until the official results are announced by the election commission. During this time, observer nodes should stop processing new transactions and handle the chain requests only.

5.5.4 Counting the votes

Once the voting is finished, any individual can count the votes by reading the blockchain. They first need to validate the chain according to the CryptoVote chain validation rules (see: section 5.5.3).

Since the private key of the election is announced by the election commission after the voting, each vote in the blockchain can now be decrypted using the private key of the election.

$\boldsymbol{I_v \mid CandidateName \mid N_r = decrypt(S_e , V_{cipher})}$

The vote is counted as valid if

1. The transaction is valid according to the transaction validation rules (see: section 5.5.3).
2. The candidate name in the vote is one of the valid candidates.
3. The key image in the vote – $\boldsymbol{I_v}$ is the same as the key image in the transaction.
4. The random number – $\boldsymbol{N_r}$ in the vote is in the range of $[1, 10^9]$.

5.6 Improving privacy by using Tor anonymity network

Although the ring signatures cryptographically helps us to protects the voter's identity from being revealed, the source IP addresses of the voters can still be tracked and it can reveal their identities. In order to avoid this scenario, we propose to use the Tor anonymity network for the communication. Using the Tor network will help us to make the tracking of the voter's IP addresses hard.

5.7 Evaluation

Eligibility

We assumed that the election commission prepares the list of the eligible voters by registering the voters' public keys with an ID card verification. Since the mapping of the public keys with the actual names of the voters are going to be published online by the election commission, it is very unlikely that the election commission would try to add non-eligible voters to the list. In order to have more transparency over this process, the registered voters can be categorized within the registration stations. Besides from the registration, the transactions are also validated against the ring signature and the eligibility of the ring signature members. These properties of the protocol guarantee that non-eligible voters cannot participate in the election.

Single-vote

The ownership of the vote is represented using the key images in the protocol. The key image is a part of the ring signature and it is unique to each voter. As a part of the transaction validation and the chain validation, we check the uniqueness of the key image in the chain. These properties guarantee that only an eligible voter can vote, and he can vote only and only once.

Anonymity

Using the key images to represent the ownership of the vote makes it extremely hard and practically impossible to find out the actual owner of the vote. Since the key image of the voter is known to the voter himself, he can always verify his vote in the blockchain. In order to protect the voter's identity from being revealed by tracing his IP address, we use the Tor anonymity network to avoid anyone to track the originating address of the requests.

Voter verifiability

Using the ring signatures allow us to achieve the voter verifiability without revealing the actual identity of the voter (see: section 5.5.2). Ring signatures allow a verifier to verify the vote by checking the eligibility of the ring members, without knowing the identity of the real voter.

Fairness

Since the voters are encrypting their votes using the public key of the election, it makes it very expensive or practically impossible to decrypt the votes in the blockchain without the private key of the election. This prevents the partial results to be counted and have influence over the voters. Once the voting is finished, the private key of the election is published by the election commission for being able to count the votes.

Integrity and correctness

A decentralized nature of the blockchain guarantees that once the vote is recorded, it becomes immutable, and no one has a central authority over the infrastructure to be able to violate the integrity.

In case any node tries to delete some votes in his copy of the blockchain, the chain will be broken as the node cannot change the previous block field of the next block. Therefore, any kind of modification will make the blockchain invalid. This doesn't prevent the observer node to build the whole blockchain by generating the hash of each block though. In that case the length of his blockchain will be smaller and if there is at least one fair observer who provides a longer valid chain, the attackers chain will be considered invalid by the network members.

Additionally, it is impossible for anyone to generate a valid ring signature without having the private key of the one of the ring members. It prevents attackers to provide a longer chain by adding invalid votes to the chain.

To generalize, having at least one fair observer would guarantee the integrity and the correctness of the election.

Universal Verifiability

In order to count the votes, one should know the private key of the election and have a read access to the blockchain ledger. In the protocol, the read access to the blockchain ledger is available to everyone, and the private key of the elections is announced after the voting is finished. Everyone is able to count the votes as described in the section 5.5.4.

Availability

A decentralized and distributed nature of the blockchain provides us highly available system. In case any observer node goes down, there are always other nodes to handle the requests.

5.8 Performance

We have tested our proof of concept implementation [24] of the protocol with several inputs in a machine with 2 GHz Intel Core i5 processor. Although the time of making a vote increases linearly as we increase the number of the ring members, time to count the votes remains quite fast.

In figure 14, we can see a linear increase in the comparison of each vote duration as the number of ring members increases. In the figure, the X-axis represents the number of the ring members, and the Y-axis represents the duration of each vote in seconds.

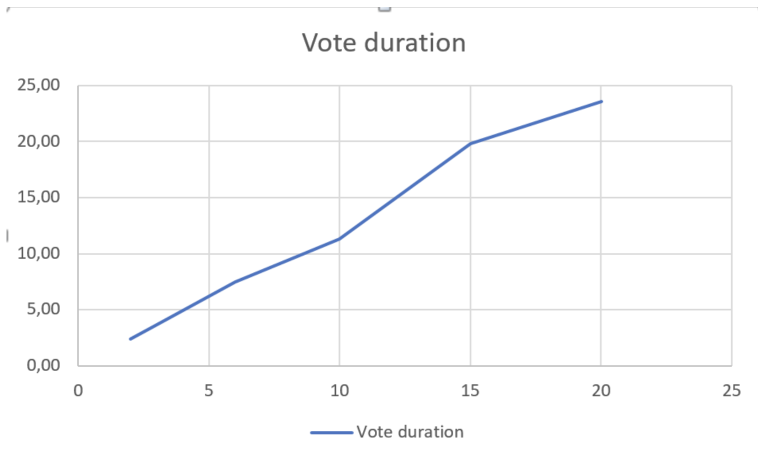

Figure 14: A single vote duration as the number of ring members increases.

Figure 15 represents the counting duration for 1000 votes as the number of the ring members increases. In the figure, the X-axis represents the number of the ring members, and the Y-axis represents the time taken to count the results in seconds. We can see that duration of counting 1000 votes remains around 1 second all the time. Therefore, we can say that counting of the votes, does not depend on the number of the ring members.

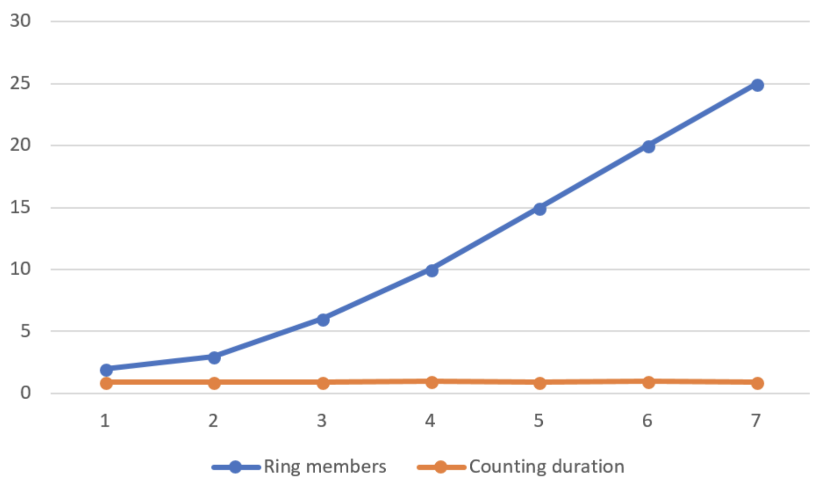

Figure 15: Comparison of the counting 1000 votes as the number of ring members increases.

Although the statistics show that the duration of counting does not depend on the number of the ring members, it does depend on the number of the participants linearly. In figure 16, the X-axis represents the number of the voters, and the Y-axis represents the duration of counting results in seconds.

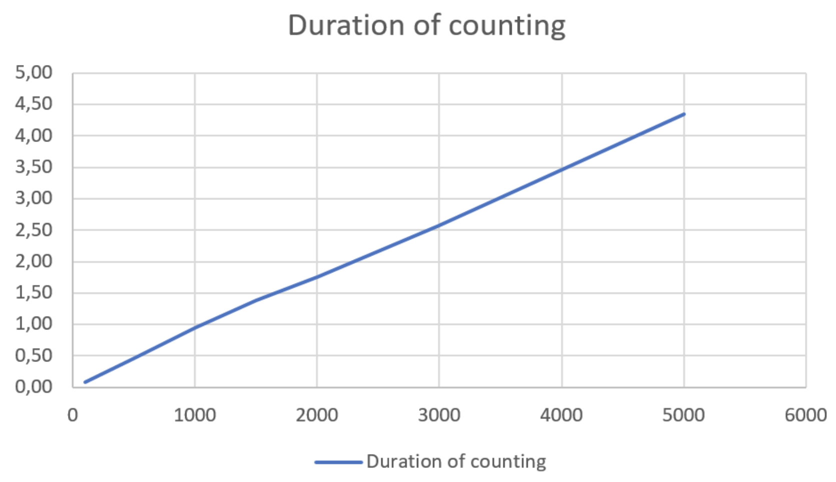

Figure 16: Results counting duration as the number of voters increases.

5.9 Conclusion

In this thesis, we proposed an election protocol where voters can vote anonymously without revealing their identity, at the same time any person can verify the correctness of the election. We solved the challenging problem of enabling transparency and protecting voters' privacy at the same time by using the public-key cryptography, one-time ring signatures and blockchain.

6 References

[1] Wilhelm, Anthony, Digital Nation: Towards and Inclusive Society, MIT Press, 2004, pages 67-73.

[2] E. G. Arnold, History of Voting Systems in California, California Secretary of State Bill Jones, June 1999.

[3] Rebecca T. Mercuri, A better ballot box, IEEE Spectrum, Oct. 2002; pages 46-50.

[4] Cronin, Mary J, Banking and Finance on the Internet, John Wiley and Sons, 1997, ISBN 0-471-29219-2 page 41.

[5] Evan I. Schwartz, Digital Cash Payoff, Dec. 2001, MIT Technology Review.

[6] Nakamoto, Satoshi, Bitcoin: A Peer-to-Peer Electronic Cash System, Oct. 2008

[7] Cuthbertson, Anthony, Bitcoin now accepted by 100,000 merchants worldwide, International Business Times, Feb. 2015.

[8] Raval, Siraj (2016). "What Is a Decentralized Application?". Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly Media, Inc. pp. 1–2. ISBN 978-1-4919-2452-5.

[9] Vitalik Buterin, August 7th, 2015, On Public and Private Blockchains, Ethereum Blog

[10] Karl Wüst, Arthur Gervais, Do you need a Blockchain?, IACR Cryptology ePrint Archive, 2017.

[11] 1363-2000 - IEEE Standard Specifications for Public-Key Cryptography

[12] How to leak a secret, Ron Rivest, Adi Shamir, and Yael Tauman, ASIACRYPT 2001.

[13] CryptoNote v 2.0, Nicolas van Saberhagen, October 17, 2013.

[14] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. J. Cryptographic Engineering, 2(2):77–89, 2012.

[15] Kristin E. Lauter, Katherine E. Stange, The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences, 2008.

[16] Fagoyinbo, Joseph Babatunde (28 May 2013). The Armed Forces: Instrument of Peace, Strength, Development and Prosperity. AuthorHouse. ISBN 978-1-4772-2647-6. Retrieved 29 August 2014.

[17] Yi Liu, Qi Wang, An E-voting Protocol Based on Blockchain.

[18] Ahmed Ben Ayed, A conceptual secure blockchain based electronic voting system.

[19] Pavel Tarasov and Hitesh Tewari, The Future of E-Voting

[20] Yifan Wu, An E-voting System based on Blockchain and Ring Signature

[21] Blockchain for digital identity. Unlocking decentralized, digital identity management through blockchain, IBM Blockchain Blog

[22] Blockchain and Smart Contracts Could Transform Property Transactions - CFO Journal. - WSJ

[23] Fritz Bauspiess, Frank Damm, Requirements for cryptographic hash functions, September 1992, 10.1016/0167-4048(92)90007-E

[24] CryptoVote PoC implementation, Valeh Hajiyev, https://github.com/valeh/cryptovote

[25] Design of Hashing Algorithms, Josef Pieprzyk , Babak Sadeghiyan, ISBN-13: 978-3540575009

[26] Announcing the first SHA1 collision, Google Security Blog

[27] World Wide Web Consortium (W3C), DSig 1.0 Identifiers