UNIVERSITY OF BIRMINGHAM

MASTER'S THESIS

Evaluation and Improvement of Two Blockchain Based E-voting System: Agora and Proof of Vote

Author:
Shuang WU

Supervisor:
Dr. David Galindo

Cyber Security
School of Computer Science

September 11, 2018

Abstract

Evaluation and Improvement of Two Blockchain Based E-voting System: Agora and Proof of Vote

by Shuang WU

In this thesis, we evaluated two existing e-voting systems based on blockchain technology: Agora and Proof of vote. We found that they both use mixnet to achieve ballot anonymity, which is inefficient and the privacy will be broken if the nods in mixnet collude. To improve this weakness, we analyzed the previous protocols and made a comparison of them and then introduced a new method that relies on the Zerocoin protocol to achieve the ballot anonymity. Based on this method, we built the **Let's Vote** e-voting system. We evaluated the performance of the anonymization of the **Let's Vote** e-voting system, the result represents that the protocol that we introduced is more efficient.

Acknowledgements

I would like to thank my supervisor Dr. David Galindo, he is very knowledgeable, well organized during the project and eager to help me all the time. I also want to thank my parents for supporting me to study here. Finally, I would like to thank Ruijia Li for giving me many useful ideas about my project, also Georgios Vasilakis who helped me a lot with the coding.

Contents

Abstract i

Acknowledgements ii

1 Introduction 1
1.1 Outline of this project . . . 1

2 Background 2
2.1 E-voting . . . 2
2.1.1 Voting privacy . . . 2
2.1.2 Vote Verifiability . . . 2
2.2 Blockchain in e-voting systems . . . 3
2.2.1 Blockchain categories . . . 3
2.2.2 Bitcoin . . . 3
2.2.3 Zerocoin . . . 3

3 Agora and Proof of vote Evaluation 6
3.1 Before voting . . . 6
3.2 Voting phase . . . 7
3.3 Ballot decryption and tallying phase . . . 8
3.4 Blockchain network . . . 8

4 Let's Vote - an E-voting System Based on Zerocoin 12
4.1 Literature review of the anonymization methods . . . 12
1.Why to choose public accumulator ZKSoK(zero-knowledge proof signature)? . . . 12
2.Do the problems that faced by Zerocoin affect the security of the e-voting system? . . . 14
4.2 Let's Vote protocol . . . 15
4.2.1 Cryptographic building blocks . . . 15
4.2.2 Assumption . . . 17
4.2.3 Let's Vote Construction . . . 17
1.Before Voting . . . 17
2.Voting . . . 20
3.Ballot decryption and tallying . . . 20
4.Audit . . . 20
4.3 Let's Vote Properties . . . 20
4.4 Zero-Knowledge Proof Construction . . . 21

5 Implementation of the Anonymization 24
5.1 Specification . . . 24
5.2 Core Function . . . 24
5.3 **Let's vote** GUI for voter . . . 26

5.4 Evaluation of the anonymization 29
5.4.1 Signature generation,verification time and signature size 29
5.4.2 Accumulator and witness generation time 31

6 Conclusion **32**
6.1 Conclusions 32
6.2 Future work 32

A **33**
A.1 Git project links: 33
A.2 Git project content: 33

Bibliography **34**

List of Figures

2.1 Instance Bitcoin transaction 4
2.2 Null data transaction 4
2.3 Two example block chains 5

3.1 Authentication and authorization of the voter 7
3.2 Agora blockchain network layers 9
3.3 Shrding mechanism 10

4.1 Mixnet 13
4.2 Credential and secret key (S, r) 18
4.3 Before voting phase 19
4.4 Credential information 19
4.5 Voting Process 21

5.1 Class in libzerocoin[8] 25
5.2 CoinSpend class 25
5.3 Voting GUI for voter 26
5.4 Voting step 2 27
5.5 Voting step 2:dialog box 27
5.6 Voting step 3:Ballot information on the bottom 28
5.7 Voting step 3:Verification dialog 28
5.8 Woring flow of the GUI 29
5.9 Signature generation and verification time 30
5.10 Size comparison 30
5.11 The time to generate the Accumulator and witness 31

List of Tables

3.1 Comparison and evaluation of the **Agora** and **Proof of vote** 11

4.1 Comparison of the computation of the public accumulator ZKSoK and ring signature 13

4.2 The comparison of the efficiency of ZKSoK and ring signature 14

5.1 The signature generation, verification time and signature size 29

Chapter 1

Introduction

The aim of this project is to evaluate and improve two blockchain based e-voting systems:**Agora** and **Proof of Vote**. The evaluation of the systems is based on the efficiency and security of every voting phase. We found that both of these two e-voting systems are using mixnet to achieve ballot anonymity, which is not efficient and it has faced some security issues. In order to improve this shortcoming, we created a new approach that is based on the Zerocoin protocol to realize the anonymous ballot and compared our protocol with the existing major methods, the result shows that our protocol has better scalability and it is more efficient than the previous protocols we analyzed. Based on our protocol, we built the e-voting system **Let's Vote** and implemented the anonymization algorithm of it to evaluated its performance. We supposed that our readers have advanced cryptography foundation i.e. public key encryption and decryption schemes, digital signatures schemes, hash functions, zero-knowledge proof schemes.

1.1 Outline of this project

The rest of this thesis proceeds as follows.

- In Chapter 2 we described the background of e-voting and blockchain.
- In Chapter 3 we evaluate the e-voting systems: **Agora** and **Proof of Vote**
- In Chapter 4 we do a literature review for the existing methods to achieve the anonymity of the e-voting system and then introduce a new method based on the Zerocoin protocol. We also give a concrete realization of our **Let's Vote** e-voting system.
- In Chapter 4 we describe our implementation of the anonymization algorithm and then evaluate its performance.
- In Chapter 5 we give a summary of our work and what we will do in the future.

Chapter 2

Background

2.1 E-voting

Voting is one of the cornerstones of the modern democracy. A transparent, accurate, impartial, auditable and robust voting system is needed for people to participate in a democratic community. The voting system should be able to keep voter's privacy and it should also be efficient and accessible for all eligible voters.Traditional voting systems are based on the paper ballot which not only costs a lot of human and financial capital but also fails to provide transparency and verifiability. While e-voting systems can mitigate some of the costs associated with the traditional voting systems, they are still suffering some flaws according to the researches[16] [38]. Germany has been abandoned online voting because of its security issues and vulnerability.

2.1.1 Voting privacy

Vote privacy is a significant human right and it is a crucial property to avoid bribery. Vote-buying was uncommon in the Western world until the 20th century. Over the years, researchers have gradually defined the concepts of vote privacy into the following pivotal properties[29]:

1. **Ballot Secrecy**: Voter's choice cannot be released to anyone else unless the voter tels them.

2. **Receipt-Freeness**: Voter cannot prove to a third party how she voted i.e. there is no evidence that can be used to prove a certain vote choice.

3. **Coercion-Resistance**: Even if a votes-buyer is monitoring the voter to cast a vote, the voter is still able to cast a vote for her own choice.

2.1.2 Vote Verifiability

In real-world e-voting, the voter has to implicitly trust the voting machines or voting software to encrypt her ballot and then cast it to the ballot bulletin, however, these end-points possibly have been compromised already. The verifiability properties of the e-voting systems are to guarantee the correctness of the voting ballot and tally result. Sako and Kilian defined two forms of verifiability[29]:

1. **Individual verifiability**: Every voter can verify that a ballot was cast on her behalf and record her choice.

2. **Universal verifiability** Any observer can verify that the result is truly calculated from the content of the ballot box.

End to end(E2E) e-voting systems redefine the verifiability of the voting systems on three aspects:

1. **Cast as intended**: Each voter is able to verify that the vote she cast has been correctly marked on the ballot.
2. **Recorded as cast**: The voter can verify that her ballot has been correctly recorded on the bulletin board ,which will be used to tally the result.
3. **Tallied as recorded**: The voter should have the ability to verify that the voting result from the tally centre is correct.

2.2 Blockchain in e-voting systems

Blockchain technology is used as a distributed public ledger in e-voting system. It could provide a transparent, immutable, undeletable distributed database to record the information generated during the election process. Because of its properties, it gives a potential way to solve the security issues that happened in the electronic voting systems. **Agora** and **Proof of Vote** both used blockchain as a public bulletin board. Which records the election configuration, voters' ballot, all the casted ballot, the tallying result and so on. As long as data has been recorded on the blockchain, it is accessible by everyone(with permission) but completely irreversible.

2.2.1 Blockchain categories

Blockchain can be divided into two types based on the right to access the blockchain network[30]:

- **Public Blockchain.** Anyone can run a node, send data and add a new block to blockchain without barrier as long as they follow the rules.
- **Permissioned Blockchain.** In order to access the data on the permissioned blockchain, you are required some sort of permission. In this type of blockchain network, only the permissioned nodes have the right to validate the blockchain transactions and add new blocks to the blockchain.

2.2.2 Bitcoin

The most practical use of the blockchain technology is cryptocurrencies and Bitcoin is the first one and the most widely used cryptocurrency. Users could send Bitcoins to other users on the peer-to-peer Bitcoin network without a central party that administrates the network. Transactions will be verified by the nodes in the blockchain network using cryptographic methods and recorded in a public distributed ledger i.e. the blockchain. An example of a Bitcoin transaction can be seen in figure 2.1

People can also record arbitrary data on the blockchain using the **Null data transaction** with the script OP_RETURN. The arbitrary data size is up to 40 bytes[3].An example of **Null data transaction** can be seen on figure 2.2

2.2.3 Zerocoin

There are also anonymous cryptocurrencies that allow anonymous transactions such as Zerocoin, Zcash, Monero. Zerocoin is a cryptographic protocol based on Bitcoin to allow fully anonymous currency transactions. Zerocoin protocol uses cryptographic

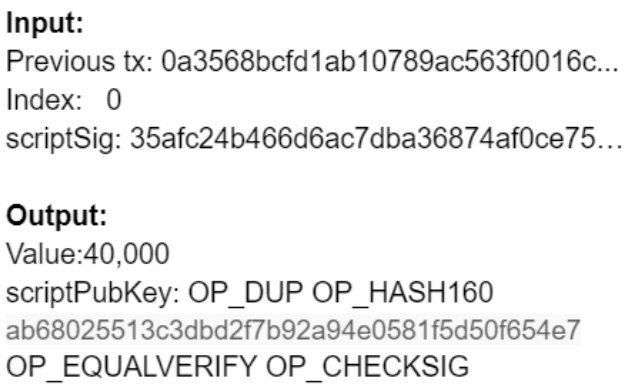

FIGURE 2.1: Instance Bitcoin transaction

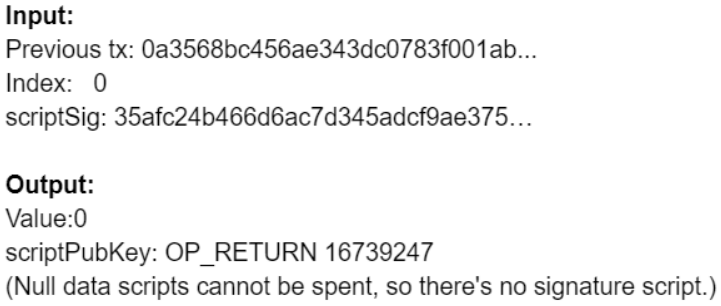

FIGURE 2.2: Null data transaction

tricks to disconnect a single Bitcoin transaction with its previous transactions without adding any other centre administrators [15].

To give a better explanation of the intuition behind Zercoin, Ian Miers et al. gave a "pencil and paper"[15] protocol instance. Provided that there is a physical bulletin board that can be accessed by all users. A user Alice wants to mint a Zerocoin of a constant value £10. Firstly, she selects a random serial number S and uses a secure digital commitment scheme to make a commitment to the serial number S. For example, if she chooses Pedersen Commitment[27] scheme, she will select another random number r, then compute the commitment as $g^S h^r$. The commitment is considered as a coin, represented by the symbol C. The only way to reveal the serial number is by using the random number r to open the commitment. Alice publishes the coin C to the bulletin board and also pins a £10 note to the announcement. All users will check the correctness of the structure of the coin, if it is correct, all users will accept it as a valid coin.

When Alice wants to spend her coin, she first collects all the valid coins on the bulletin board to get the set $C_1, ..., C_N$. Then she computes a non-interactive zero-knowledge proof π to demonstrate she is aware of the following two pieces of information:

1. A valid coin $C \in (C_1, ..., C_N)$.

2. A value r that can open the commitment C to reveal the serial number S.

The last step that Alice needs to do is using an anonymous channel(for example, using Tor[43]) to post (S, π) to the bulletin board. The other users verify the validation of the zero-knowledge proof π, which means they verify the truth that:

1. Alice indeed knows a valid coin that belongs to the coin set $C_1, ..., C_N$(however, users cannot know which valid coin it is because of the zero-knowledge proof).

2. This coin is a commitment to S.

3. Alice knows the random number r to open the commitment.

The users also check that the serial number S has never been posted before. If the (S, r) met all the conditions above, Alice can collect £10 from the bulletin board. If not, the other users decline her transaction and forbid her from getting the currency.

This protocol achieves anonymous transactions since Alice's minted coin on the bulletin board cannot be connected to her withdraw transaction (S, π). In order to know which coin that Alice spent, one must know the random number r, the zero-knowledge doesn't reveal any information about the coin she spent, so as long as Alice doesn't tell other people the random number r, the transaction can keep being anonymous. Since every serial number can only be used one time, double-spend is impossible.

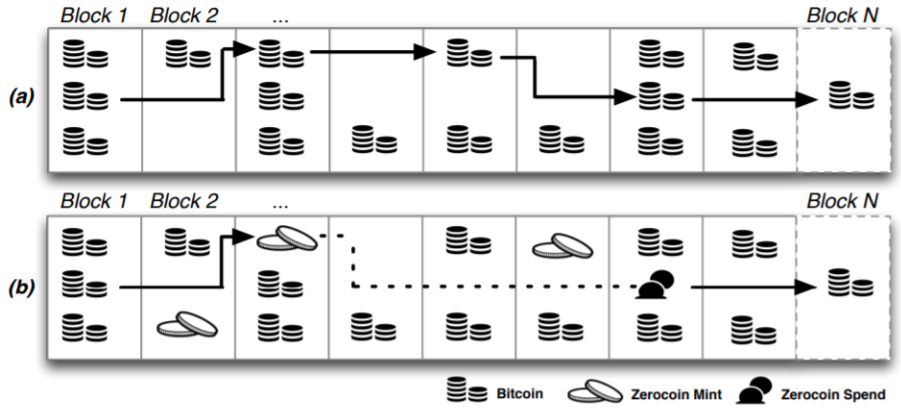

FIGURE 2.3: Two example blockchains. Chain(a) explains the bitcoin transaction history, every transaction is connected to a previous transaction. Chain(b) is Zerocoin transaction history, the connection between the minted coin and spent transaction(dotted line) cannot be distinguished by the blockchain information [15].
Image source:"Zerocoin: Anonymous Distributed E-Cash from Bitcoin"[15]

Chapter 3

Agora and Proof of vote Evaluation

The following part discusses the voting procedures and shortcomings of **Agora** and **Proof of Vote**. The analysis is mainly according to three voting processes i.e. before voting, voting phase, ballots decryption and tallying phase. We also evaluate their blockchain network at the end.

3.1 Before voting

The most important thing that e-voting systems need to do before voters casting their ballot is to define the eligible voters. The way they manage to do this is as following:

Agora: The administer of the election will create a configuration file which includes the list of all eligible voters. Depending on the situation, the list may be open or protected[40]. Voters need to show their real identity to cast their ballot on the blockchain successfully. The first problem of this mechanism is the privacy. All legal voters' identities will be seen on the blockchain. Secondly, if a voter's identity information is collected by a malicious person(which is common in the modern Cyber environment), he can easily forge a valid ballot. Namely, if the authority of the election official doesn't give every voter a unique secret token for the eligible check, the integrity of the voting ballot is easy to be broken.

Proof of Vote:This e-voting system requires three steps before a voter is able to cast his or her ballot[32].

1. **Generate an RSA key pair**. Hash the public key as the voter's pseudonym. The pseudonym is regarded as the identity of a voter to participate in the voting phase.

2. **Authentication**.A voter sends his/her real identity information along with his pseudonym to the authentication centre. The authentication centre returns a signature of the voter's pseudonym. The purpose of this process is to combine a real person with a pseudonym.

3. **Authorization**.This is the process by which an authenticated voter is authorized to vote. It operated similarly to a registration process, namely, give the right of election to a pseudonym that has already be linked to a real voter.

Compared to Agora, Proof of Vote is safer and more private in terms of before voting phase, but it requires more participation from voters. Apart from the complexity, the voting authentication centre has the ability to link an encrypted ballot to a real person because the voter will used her pseudonym to cast a ballot, while the authentication centre knows who is the owner of this pseudonym.

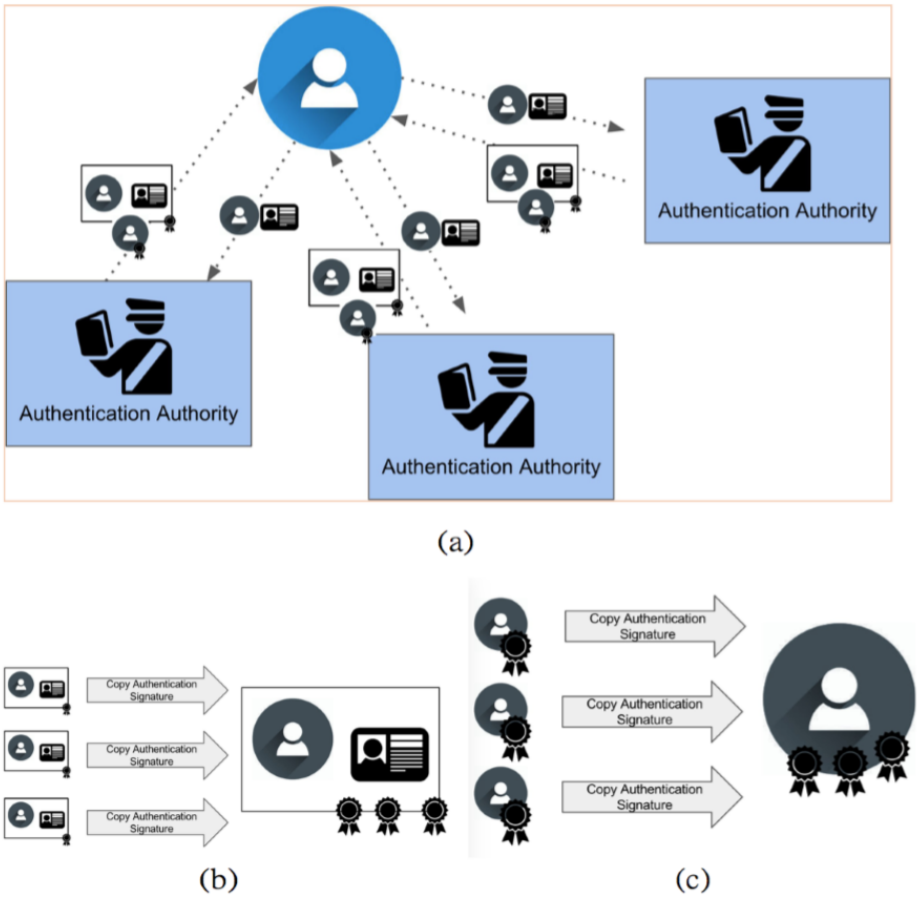

FIGURE 3.1: Authentication and authorization of the voter: (a) voter sends Pseudonym, Identifying-information, and Proof-of-Identity to a quorum of authenticators. Authenticators send back a signed pseudonym and a signed package comprised of the pseudonym and Canonical Identifying-Information. (b) Voter consolidates all the authentication signatures in the package containing pseudonym and Canonical Identifying-Information. (c) Voter consolidators all authentication signatures on the standalone pseudonym.
Image is cited from "Proof of Vote White Paper" [32]

In Chapter 3. I will describe a new e-voting system that combines the advantages of **Agora** and **Proof of Vote**, which means a voter will receive a unique secret token from the election official but without a registration step needed before voting.

3.2 Voting phase

After the election official finishes the voting configuration, the eligible voters start to send their ballot. We analysis two aspects in this process: How they achieve cast as intended and how they realize the anonymity of the ballot.

- **How they achieve cast-as-intended**. Both **Agora** and **Proof of Vote** use the threshold ElGamal algorithm to encrypt the voter's ballot and allow voters perform a Cast-or-Challenge validation[2].

- **How they realize the anonymity of the ballot**. They both use Mixnet to achieve the anonymity of the ballot. This mechanism gives a way to destroy ballot privacy if enough trustees collude. Because the trustees jointly hold the decryption key, they can collude to decrypt the encrypted vote before the votes are anonymized by the mixnet. More precise description of the mixnet is in the next chapter.

The ElGamal encryption will be described in the next chapter. Mixnet is generally inefficient, we created an anonymization protocol based on Zerocoin protocol that is more efficient than mixnet, the details will be discussed in the next chapter.

3.3 Ballot decryption and tallying phase

In order to perform the tallying phase, both **Agora** and **Proof of Vote** need at least a specific number of trustees to cooperate to decrypt the ballots since they both use threshold ElGamal encryption to encrypt the ballots. The trustees will publish their partial decryption result with the correctness proof onto the blockchain for the tallying process.

To start this process, firstly every trustee checks the validation of the zero-knowledge proof that the mixnet servers generated are correct, and if so, the trustees will partially-decrypt every anonymized ballots and generate a zero-knowledge proof that is used to verify they did the decryption correctly. Once finished, all trustees send their results to the blockchain.

Supposed that there are enough trustees generated the correct partial decryption i.e the number of honest trustees is equal or bigger than the threshold of the ElGamal encryption. The election official can use these partial decrypted ballots to recover the plaintext ballots, this process can also be done by anyone who can access the data on the blcokchain. The plaintext ballots will be posted to the blockchain where they can be tallied.

After the trustees decrypt the ballots, any user of the blockchain is able to tally votes across all plaintext ballots. The election official should publish the final result.

3.4 Blockchain network

Agora and **Proof of Vote** both are based on blockchain technology to obtain an immutable record of all data throughout the whole election process. However, their blockchain architectures are different.

Agora blockchain network has three layers[40]: the bulletin board, Cotena, and the Bitcoin blockchain. The bulletin board is a permissioned blockchain network that has write-permissioned nodes run by Agora, witness nodes that operated by recognized third-party and read-only nodes that can be operated by any user. The bulletin board blockchain is based on Skipchain architecture[23], which is a novel data structure that provides a proactive Byzantine consensus mechanism. The skipchain data structure enables the software client to get a proof of transaction validation with no need to download a full record of the blockchain data and to efficiently look through any length of the blockchain timelines both backward and forward.All the information that generated during the election will be recorded on the bulletin board blockchain.

The permissoned nodes in the bulletin board blockchain network exchange information and manage the second layer: **Cotena**, it is originally from Catena scheme[42].

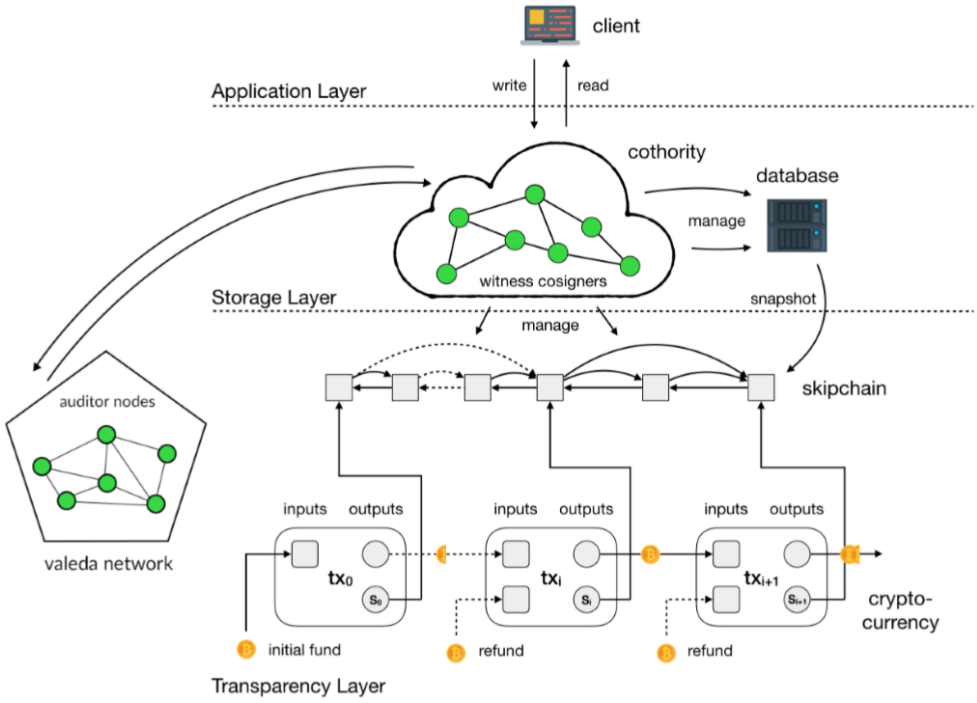

FIGURE 3.2: Agora blockchain network layers
Image source: "Agora white paper version0.2"[40]

Built on top of the bitcoin blockchain, Catena provides a tamper-resistant logging mechanism. It will periodically take a snapshot of the bulletin board and record the snapshot log both on permissoned nodes and Bitcoin blokchain via the *OP_RETURN* opcode.

During an election, the updated election information can be recorded to the bulletin board blockchain very frequently(e.g., once a minute). Cotena takes the snapshot log of the bulletin board in a less frequency(e.g. once a day), then this log is updated to the Bitcoin blockchain for decentralized immutability and transparency.

The skipchain architecture enables high throughput and efficient transaction validation, therefore it is capable for resource-constrained clients. The Cotena logging mechanism helps the data storage in **Agora** system to be more secure since it is recorded to the Bitcoin blockchain.

The **Proof of Vote** blockchain network operates on a proof-of-authority[31] basis. The central authority gives permission to the blockchain nodes to allow them to operate the blockchain. The nodes include different parties such as government, election administrators, election participation and so on. In order to decrease the latency of transaction and confirmation time for each block, only a limited number of nodes can propose and vote on blocks[32].**Proof of Vote** also introduce a Sharding[12] mechanism to improve the throughput and decrease the latency.

In order to prevent the double spending, traditional blockchain requires every node to maintain all the history blocks, this can pose a challenge when people want to scale the throughput of the network. The sharding mechanism allows transactions to be recorded either to the main chain or to a sidechain. Because every voter in **Proof of Vote** system has a unique RSA key pair, a voter needs to hash her public

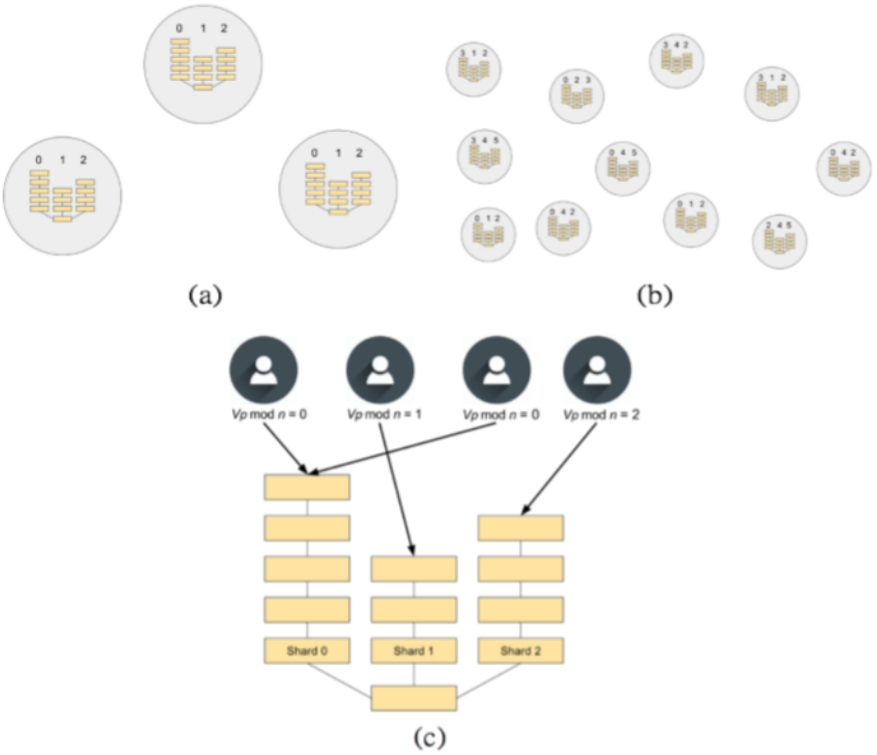

FIGURE 3.3:
Sharding mechanism of the Proof of Vote: transactions can be recorded to the different sidechains.
Image source:"Proof of Vote while paper" [32]

RSA key as her unique pseudonym and provide this information when generate her ballot, therefore the double-spending problem is to be prevented on a per-voter basis. The validity of one vote is associated with other votes' validity, where, more specifically, when a voter post her vote on blockchain network, all nodes will check if her pseudonym has been used before or not, the transaction will be accepted only the pseudonym hasn't been used before.

Though the Sharding mechanism can improve the transaction latency and throughput on a blockchain, it requires more work when nodes validate the votes. Compared to **Agora** that leverage the data security of Bitcoin blockchain, the **Proof of Vote** has less conviction for their security of the blockchain network.

The table 3.1 shows the comparison of the properties of **Agora** and **Proof of Vote**.

TABLE 3.1: Comparison and evaluation of the **Agora** and **Proof of vote**

Category	**Concern**	**Agora**	**Proof of vote**
Voter Anonymization	Anonymity	Mixnet	1.Mixnet 2.Pseudonym
	Ballot privacy	Threshold ElGamal encryption	Threshold ElGamal encryption
	Collusion of key holders	×	×
	Forward election integrity	trustees destroy private keys	trustees destroy private keys
Voter Authentication	only eligible voter cast votes	Listing all eligible voters on blockchain	RSA key pair for every voter
Verification	Cast as intended	Benaloh challenge[2]	1.Benaloh challenge[2] 2.Decentralize challenge(using second device)
	Recorded as cast	1.Blockchain record 2.mixnet zero knowledge proof	1.Blockchain record 2.mixnet zero knowledge proof
	Counted as recorded	Blockchain record	Blockchain record
Receipt freeness		×[1]	×[1]

[1] The receipt-freeness will be broken if the voter gives her randomness that used in ElGamal encryption.

Chapter 4

Let's Vote - an E-voting System Based on Zerocoin

In this chapter, we propose a new method based on Zerocoin to achieve the anonymity property of e-voting systems. Based on this method, we built the fully anonymous e-voting system **Let's Vote**. In the first part, we do a literature review of the major approaches that can achieve anonymity in an e-voting system. In the following part, we describe the concrete instantiation e-voting system **Let's Vote**.

4.1 Literature review of the anonymization methods

The anonymity of **Let's Vote** e-voting system is realized by using the public accumulator proposed in [6], it is used by Zerocoin[15] to allow fully anonymous currency transactions. But the public accumulator has never been used to achieve the anonymity property in an e-voting system. In the following part we compare the performance of the public accumulator with other anonymization protocols and then present the advantages of this protocol. Zerocoin protocol faces both efficiency and security issues in practice, we gave an analysis that these problems won't affect the e-voting system.

1.Why to choose public accumulator ZKSoK(zero-knowledge proof signature)?

For many years, there are many different methods were proposed to achieve the anonymity of e-voting systems. Below is the review of the major ones.

1. **Blind signature.** [7] [11][24][25]: In e-voting systems based on blind signature algorithm, voters can get the authentication signature on their ballot without leaking the content of their ballot. Nevertheless a registration phase is necessary for every voter before each voting event. Meanwhile, the administrator has the ability to add fake ballots to bias the tally in the registration phase.So the e-voting system doesn't have the universal verifiability[33].

2. **Mix-nets.**[26] [34] [17] [40] [32] [28] [19] [14] The Mix-net used in e-voting systems is a network of servers that re-encrypt and shuffle the votes.The input ballots order cannot be matched to output order. The mix-net in general is not efficient because of several rounds of re-encryption and it needs to generate the proof of correctness of the encryption. Furthermore, the mistakes in the design of the protocol or the implementation of the protocol will potentially allow mixnet nodes have the ability to individually compromise the integrity of the encrypted ballots. Additionally, if a sufficient number of nodes collude, they can destroy the ballot privacy.

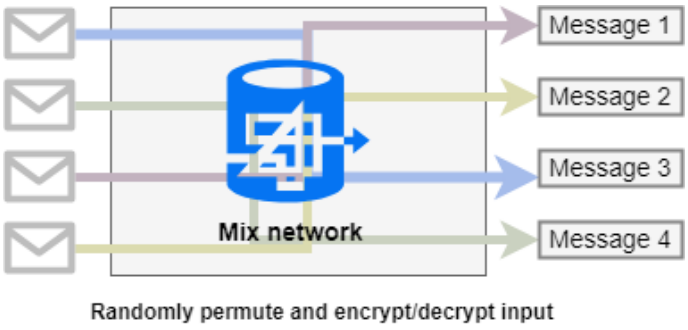

FIGURE 4.1: Mixnet

The **Agora** and **Proof of vote** both use mixnet to anonymize the ballot, which is the main weakness that we are going to improve.

3. **Ring signature.**[20] [44] [1] [45] [13] Ring signature is a group signature scheme without the group administrator. Namely, a ring signature could be recognized that it is generated from a group of people but no one could distinguish who in this group actually signed the signature. The group size defines the anonymity level, the more people are included in the ring signature, the higher anonymous level the signature has. However, with the increase of the group size, the signature size and the time to generate and verify the signature increase significantly.

TABLE 4.1: Comparison of the computation of the public accumulator ZKSoK and ring signature

Algorithm	**Sign computation**	**Verify computation**
liu et al.[20]	$2(n-1)M+3E$	$2nM$
Tsang and Wei[44]	$(2+n)E+7M$	$7M$
Au et al.[1]	$(2+n)E+7M$	$7M$
Zheng et al.[45]	$(14n+2)$ $seq.op.$	$(14n+2)seq.op.$
Fujisaki [13]	$(6+13\sqrt{n})E$ $+(5+n+2\sqrt{n})M$ $+2\sqrt{n}P+OTS$	nM $+(8+2n+12\sqrt{n})P$ $+OTV$
Our protocol	$kE+15E$	$4kE+25E$

[1] The computation results of all the ring signatures in the table is cited from [35]

[2] According to [35], they used E and M to indecate an exponentiation and a multi-bases exponentiation respectively, P is used to denote a pairing. OTS is used to represent a one-time signature signing and OTV is to indicate a one-time signature verification.

[3] The n means the anonymity group size(the number of voters that included in one ring signature)

Table 4.1 compares the computation of $ZKSoK$ and several ring signatures.As the table shows, the computation of the ring signatures depends on the anonymity group size n, while the computation of $ZKSoK$ depends on the predefined constant security level k.The computation of $ZKSoK$ won't change by the anonymity group size.

TABLE 4.2: The comparison of the efficiency of ZKSoK and ring signature

Algorithm	**Anonymity group size**	**Generation time(ms)**	**Verify time(ms)**
Our protocol	4	233	197
Liu and Wong[21]	4	82	75
liu et al.[20]	4	232	247
Our protocol	200	233	197
Liu and Wong[21]	200	21671	13265
liu et al.[20]	200	463644	465929
Our protocol	500	179	139
Liu and Wong[21]	500	159898	84509
liu et al.[20]	500	3060601	9840218

[1] The source code of the ring signature included in the table can be found in [37]
[2] The experiments were conducted on an Intel(R) Core(TM) i5-6200U CPU(2.3GHz 2.4GHz) with 8GB of RAM, running 64-bits Windows 10.

We run two implements of the light ring signature to make a comparison of their efficiency with our protocol. Table 4.2 shows the time to sign and verify a signature that generated by the different signature scheme. The comparison is based on three different anonymity group sizes.

With the analysis above, we conclude that the public accumulator $ZKSoK$ has the following advantages:

1. **Less participation.** Before voting, a voter only needs to send her personal identity information to the authentication centre to get her unique token,namely, the credential and its corresponding secret key, no registration is needed.

2. **Efficiency.**The size and efficiency of the algorithm i.e. the generation and verification time of the ZKSoK depends on the security level λ, rather than the anonymity level. In this respect, the public accumulator is more suitable for large-scale voting to achieve high-level anonymity.

2.Do the problems that faced by Zerocoin affect the security of the e-voting system?

Efficiency issue Among other anonymous cryptocurrencies like Monero coin and Zcash, Zerocoin has the downside that it is inefficient. Zerocoin authenticates the valid coins by using zero-knowledge proof that requires double-discrete-logarithm proofs. The size of the proof exceeds 45kB and require 450ms to verify[9]. However, in e-voting systems, this problem can be mitigated by the following two aspects.

- Firstly, in e-voting systems, we always need an administrator,namely, the election official. It helps the user to compute some public parameters. while in Zerocoin ,all these values can only be calculated by the users themselves since there is no centre in the blockchain network. By this way, the signature generation time could be minimized up to 30%.

- Secondly, Monero coin uses the ring signature to achieve anonymous transactions, but the recommended anonymity group size is only $4 \sim 10$[22]. Obviously, this anonymous group size is not enough if it is used for an e-voting

scenario. When the anonymous group size goes up to 100, the signature size and the verification time will be huge. However, Zerocoin is based on the accumulator and its performance won't change by the anonymity group size. The anonymity group size could easily go up to thousands without any additional computation burden added to the user's device. Therefore zerocoin is more efficient in an e-voting scenario. Actually, there is research on accumulator-based ring signature that is going to be used in the future version of Monero[39].

Security issue On April 2018, Tim Ruffing et al. published the paper "Burning Zerocoins for Fun and for Profit"[41] which describes a security issue in the design module of the Zerocoin. They describe that if a malicious person could intercept the coin information when a user is going to spend her coin, the malicious person could make the coin invalid. Then the user won't spend her coin successfully. However, this security issue won't affect the design of an e-voting system because in Zerocoin the attack is based on the dynamism of the *Accumulator*(one of the parameters in the algorithm) while the *Accumulator* in e-voting system is static.

4.2 Let's Vote protocol

To understand the intuition behind the *Let'sVote* e-voting protocol, we go over the "pencil paper" protocol[15] that we mentioned in Chapter 1. An e-voting variant is as follows: Alice wants to cast a ballot, she first choose a random serial number S and makes a commitment to it. The resulting commitment is a credential, denoted as C. Alice sends her credential along with her identity information to the election official. The election official checks the eligibility of Alice according to her identity information, if Alice is eligible to vote, the election official will publish her credential to the bulletin board. When Alice is going to vote, she will collect all the credentials published by the election official, then produce a zero-knowledge proof signature π on her ballot information m. The zero-knowledge proof is to prove that she knows a valid credential which is a commitment to S. She posts the (S, π) along with her ballot information to the blockchain network, the nodes will check the validation of the proof π and check if the serial number S has been used before. If it met the condition, the nodes will record her ballot on the blockchain. In practice, some parameters that Alice needs to produce the zero-knowledge proof is computed by the election official for the sake of less computation consumption in the user end.

4.2.1 Cryptographic building blocks

1. Zero-knowledge proofs and signatures of knowledge.
The zero-knowledge protocol is used to prove that a prover knows a secret but without releasing the secret to the verifier. The constructions of **Let's Vote** use the same zero-knowledge proof and signatures protocols with these used by zerocoin[15].

The notation of Camenisch and Stadler will be used to denote these proofs. For instance, $NIZKPoK\{(l,k) : h = g^l \wedge c = g^k\}$ means a non-interactive zero-knowledge proof of knowledge (l,k) such that $h = g^l$ and $c = g^k$. All variables not included in ()'s are supposed to be known to the verifier. In the same way, $ZKSoK[M]\{(l,k) : h = g^l \wedge c = g^k$ denotes a signature of knowledge on message M.

2. Accumulators.

Accumulator used in the **Let's Vote** e-voting system is proposed by Camenisch and Lysyanskaya [18]. The accumulator is based on the Strong RSA assumption.In zerocoin, the accumulator is computed by the node of the blockchain, while in the e-voting system, the accumulator is calculated by the election official.

The accumulator comprises the following algorithms[15]:

- $AccuSetup(1^\lambda) \rightarrow params$. Input a security parameter λ,select primes p, q(with polynomial dependence on the security parameter). Calculate $N = pq$, and select a seed value $u \in QR_N, u \neq 1$. The Output is (N, u) as $params$.
- $Accumulate(params, \mathbf{C}) \rightarrow A$. Input $params$ (N, u) and a set of numbers $\mathbf{C} = \{c_1, c_2, ..., c_i \mid c \in [\mathcal{A}, \mathcal{B}]\}$[1]. The accumulator A is computed as $u^{c_1 c_2 \cdots c_n} \ mod \ N$.
- $GenWitness(params, v, C) \rightarrow w$. Input $params$ (N, u) and $\mathbf{C}$ as mentioned above, a value $v \in \mathbf{C}$, compute w as $w = Accumulate(params, \mathbf{C} \backslash \{v\})$
- $AccVerify(params, A, v, w)$. Input $params$ (N, u), an value $v \in \mathbf{C}$, an accumulator A and a witness w, calculate $A' \equiv w^v \ mod \ N$. if $A' = 1$, output 1,else output 0.

Camenisch and Lysyanskaya proposed an efficient zero-knowledge proof of knowledge that a committed value is included in an accumulator. Ian Miers, Christina Garman etc. converted it into a non-interactive proof according to the Fiat-Shamir transform[10][15].The zero-knowledge proof uses the following notation:

$$NIZKPoK\{(v, w) : AccVerify((N, u), A, v, w) = 1\}$$

4. RSA signature

The RSA signature is used by the election official to authenticate the voters' information on the blockchain. The security of the scheme depends on the hardness of factoring large numbers. The algorithm is as follows:

- Choose two large random prime numbers p, q such that $p = 2q - 1$, compute $n = pq$.
- Select an integer $e > 1$ such that $gcd(e, \varphi(n)) = 1$ where $\varphi(n) = (p-1) \cdot (q-1)$
- Find a d to satisfy the equation $e \cdot d \equiv 1 (mod \ \varphi(n))$.
- Publish the public key $PK = (e, n)$,retain the secret key $SK = d$.

For a message $M, M \in 1, 2, ..., n-1$. let h be a SHA-256 hash function, then we compute the signature by

$$s = h(M)^d \ mod \ n$$

Given the public key $PK = (e, n)$, verify the signature by comparing:

$$h(M) \qquad \text{with} \qquad s^e \ mod \ n$$

4. Threshold ElGamal encryption

The **Let's Vote** uses threshold ElGamal encryption to encrypt the ballot. The algorithm has distributed key generation and the private key can only be extracted with the cooperation of a threshold number of trustees that holds the distributed private key.

The threshold ElGamal encryption algorithm can be described as following:

1. Let $\mathbb{G}$ be a multiplicative group, the order of the group is a prime number q. p and q are big prime number and they satisfy $p = 2q + 1$, g is the generator of the group $\mathbb{G}$. There are w trustees $P_1, P_2, ...P_w$ cooperating to manage the encrypted ballots.

2. Similar to the normal ElGamal encryption, every trustee has a secret key $x_i \in Z_q$. The public key of the P_i is $y_i = g^{x_i} \ mod \ p$. The public key of the ElGamal encryption is $y = \prod_{i=1}^{\omega} y_i \ mod \ p$, the secret key is $s = x_1 + x_2 ... + x_\omega$.

3. The voter uses public key y to encrypt the candidate information m to obtain the cipher pair (g^r, my^r). In order to achieve t-threshold ElGamal encryption i.e. the decryption could be done by the cooperation of t trustees, every trustee needs to share its secret key x_i through Shamir (t, N_T)-threshold secret sharing scheme[36].

   - The trustee P_i constructs a polynomial :
   $$F_i(x) = \sum_{k=0}^{t-1} f_{i,k} x^k \ mod \ q \quad \text{where} \quad f_{i,0} = x_i.$$
   - The trustee P_i compute $d_{i,j} = (j, F_i(j))$ for every other trustee j, $j = 1, 2, ..., i-1, i+1, i+2, ..., w$, this message is encrypted by the public key of the trustee P_j.

4. When there are t out of w trustees cooperating together, each of them compute their decryption part :
   $$s_j = g^{rd_j} \ mod \ p \quad \text{where} \quad d_j = \sum_{i=1}^{w} F_i(j) \ (mod \ q)$$
   The cipher pair (g^r, my^r) could be decrypted as:
   $$m = \frac{my^r}{\prod_{j=1}^{t} s_j^{u_j}} \ (mod \ p) \ \text{ where } \quad u_j = \prod_{k=1, j \neq i}^{t} \frac{k}{k-j}$$

4.2.2 Assumption

The security of the **Let's Vote** protocol relies on the following assumptions:

- Discrete Logarithm assumption. Given a cyclic group $\mathbb{G}$ and its generator g, for $h \in \mathbb{G}$ it is hard to find x such that $h = g^x$.
- Strong RSA assumption. Given a modulus N of unknown factorization and a random element $c \in \mathbb{Z}_n^*$, it is infeasible to compute any pair (x, e) that met the equation $x^e \equiv c \ mod \ N$.
- There is at least one honest nodes in blockchain network.
- The SHA1 hash function is secure.

4.2.3 Let's Vote Construction

1.Before Voting

Before voting starts, every voter should create her own credential that used to show the eligibility of voting.The election official collects all the valid credentials and generates the corresponding information for every credential. Afterwards, the election official sends all the data on blockchain. The following part gives a more concrete description.

Credentials collection phase The election official checks every voter's eligibility. If the voter is legal to vote,the election official will record the voter's credential to the bulletin board.The algorithm used to generate credentials is as following:

- $Setup(1^\lambda) \leftarrow params$. Once get the input security parameter λ,next run $AccuSetup(1^\lambda)$ to receive the public $params$ i.e.(N, u).Then generate two primes p, q that satisfy $p = 2^w q + 1$ for $w \geq 1$. Choose random generators g, h such that $\mathbb{G} = \langle g \rangle = \langle h \rangle$ and $\mathbb{G}$ is a subgroup of $\mathbb{Z}_q^*$. The output of the function is $params = (N, u, p, q, g, h)$
- $GeneCredential(params) \rightarrow (c, sk_c)$. Select $S, r \leftarrow \mathbb{Z}_q^*$ and calculate $c \leftarrow g^S h^r \ mod \ p$ such that $\{c \ prime \mid c \in [\mathcal{A}, \mathcal{B}]\}$. Set $sk_c = (S, r)$ and $output \rightarrow (c, sk_c)$.

The election official first run the function $Setup(1^\lambda)$ to get the $params$ and send them to all the legal voters.The legal voters next run $GeneCredential(params)$, send credential c to the election official and keep the $sk_c = (S, r)$ secret.

The instance of a valid credential and its corresponding secret keys are shown in the figure 4.2.

- **Credential:**167b9ec65aed2d3cd049cb3c18017481eaf3d4caa8a1b1a6952c7fe7e0e7 6504bb50502544ea8a7cd6f222c8f97db5463f6b562b75f9f6bbd930f4dc395d304a8e2 7f75bd942166d5746da42850b5d856ad8866858a9fa59a436760bdb388bac32196067 4c931e55d2c53cd12b7b0f8b0baccee2df0c3ac94aee265fc2d98b3b
- **S:**279b552b58ed401150cd561bacbc1804d444cedaaa541b5b5c048dc35d5da68
- **r:**92754efcab38b0f3be554a8df0b6d4109a02541dff24b82a1b419c162ef3a5c0

FIGURE 4.2: Credential and secret key (S, r)

The election official are supposed to collect all the legal voters' credentials during the voting credential collection phase. After the credential collection phase close, no one could send his or her credential to the election official any more.

Generating credentials' information phase The additional information that a voter needs in order to cast a ballot include:

- **params** $\Rightarrow$ the output of the function $Setup(1^\lambda)$.
- **Accumulate** $\Rightarrow$ the output of function $Accumulate(params, \mathbf{C})$ with the set of all the valid credentials as input.
- **Witness** $\Rightarrow$ the output of the function $GenWitness(params, v, \mathbf{C})$ with a certain valid credential and the set of all valid credentials as inputs.
- **Signature of the above information** $\Rightarrow$ The election official uses its secret key sk_e to sign the signature to protect the integrity of the information above.

All these information would be generated by the election official, while a voter could also calculate them by herself if her mobile device has the computation ability and he doesn't care about the time consumption. The information are collected using the following algorithm:

- $GenVoterCre(params, c, sk_e, \mathbf{C}) \rightarrow (A, w, \pi)$. If $c \notin \mathbf{C}$ output $\perp$. Compute $A \leftarrow Accumulator((N, u), \mathbf{C})$ and $w \leftarrow GenWitness((N, u), c, \mathbf{C})$. Generate signature on (A, w) using the secret key sk_e of the election official $\pi \leftarrow \mathbf{Sig}(A, w, c, sk_e)$.

Finally, the election official submit the resulting values (A, c, w, π) to the blockchain(bulletin board).

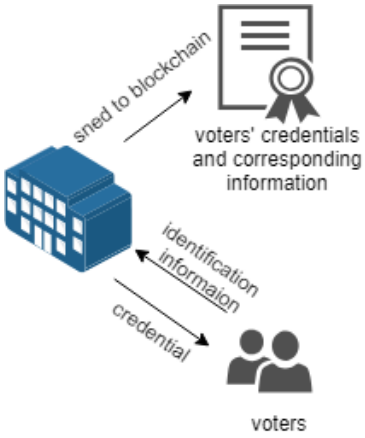

FIGURE 4.3: Before voting phase

An instance of credential's *accumulator* and *witness* is shown as in the figure 4.4.

- **Credential:**167b9ec65aed2d3cd049cb3c18017481eaf3d4caa8a1b1a6952c7fe7e0e7
6504bb50502544ea8a7cd6f222c8f97db5463f6b562b75f9f6bbd930f4dc395d304a8e2
7f75bd942166d5746da42850b5d856ad8866858a9fa59a436760bdb388bac32196067
4c931e55d2c53cd12b7b0f8b0baccee2df0c3ac94aee265fc2d98b3b
- **Accumulator:**3bf4c77ae21289b3204b335f4027fbfbebd5c24b5edeab82fa109bbd0d2
0531e92572c803d39eea7229a05befa56890c3f4e3a8b11c5e759de3cfd9175e2b15b
6068cb17d194efd2d4246369d8180962e9ac322d89e748ee783c9122128ee3939c1c
d6aeaf454f302051987fd863277ac58820e44f007464b2f0f9ca39d5a95db4e0d33aa6d
3982c77a94aa265b3eaaba99cb9a369e7678949a053b688d3c248c9bedbd2b2da996
9e0430f75c4f282a0235f960a6993b8e15f2ce7424bf21403b16f61a2d24ee24f1ecdb5
2473629f2d0143b758516e4e22a66a80f1ecccfa58ba338c56840fb67d4234e3ad9066
7c187d9942961e436585fa1433979cdaade0
- **Witness:**a0a098ed1086b1d21f7972e39bf1ed980812461d82b46236cf54a778c12dad
2da0e33e764481855b36e05f0c817c6b009b9fbfa51b4079a5466bd32c3029ccd2c825
5e46f5d3a0068fc2c6cd9a1c477b1986ef6868bdd7f464c0d295c710b8e67365f319421
b9097c60cc92a9eaec6a1d0eaebb5de7151d9094b2886236d9e85d7e583bd65e679c
51fe5bb248a2f1fcd90478d8e75ef3ba48513b8d33574360f7bfa08d7f818a5c2bddcb5
5eaf1c4f042826b6549892cedbcfc4c6be351ef7f8697d2e6ef63be0f3cd26d70349b997
ee3c0767d03da803fa0ad0269811e189ba5fc6cdb638e7669117af967f70b61f55e828
98a1a68e655240e33429a7372b0f

FIGURE 4.4: Credential information

2.Voting

A voter first collect her voting credential from blockchain. Next the voter verify the signature of the election official using the election official's public key pk_e. Then the voter should run the following functions to generate her anonymous ballot.

- $Encrypt(pk_e, \beta) \rightarrow (m_\beta)$ Input the public key of the election official y, $\beta \in \{1, ..., n\}$ is the selection of the candidate. Output voter's encrypted choice $m_\beta = (g^r, \beta y^r)$.
- $AnonymlizeBallot(params, A, sk_c, c, w, m_\beta) \rightarrow (m_\beta, S, \pi_B)$ Input the voting credential information i.e. (A, w, c), the secret of the credential i.e.(S, r), the encrypted voter's choice. Output $ballot = (m_\beta, S, \pi_B)$ where comprises the following signature of knowledge:
  $\pi_B = ZKSoK[m_\beta]\{(c, w, r) : \textbf{AccVerify}((N, u), A, c, w) = 1 \wedge c = g^S h^r\}$[1]

In the end, the voter submits the resulting ballot to the blockchain, the nodes will accept the ballot only the VerifyBallot function output 1.The following algorithm is the VerifyBallot:

- $VerifyBallot(params, \pi_B, S, R, A)$. Given a proof π_B, a serial number S, and an accumulator A, verify that π_B is the aforementioned signature of knowledge on m_β. Output 1 if the proof verify successfully, otherwise output 0.

3.Ballot decryption and tallying

After the voting phase ends, each trustee processes all the encrypted ballots recorded on the blockchain and creates a transaction that was composed of:

1. The partially decryption for every encrypted ballot is s_j, as describe in the **Threshold ElGamal encryption** above.
2. A zero knowledge proof of the correctness of the partial decryption.

After trustees decryption, anyone can independently tally the results because all encrypted ballots have been decrypted.The election official is supposed to produce the official result and send the result to blockchain.

4.Audit

All the information that generated by every step of the election will be recorded on blockchain, therefore the whole process of the election is auditable to all voters and thrid-party observers.

4.3 Let's Vote Properties

- Ballot Secrecy: All the ballots are anonymized, no one could link the ballot to the voter unless she reveal her secret key. The ballots are also encrypted by ElGamal encryption.

[1]See the last section **Zero-knowledge proof construction** for more precise description

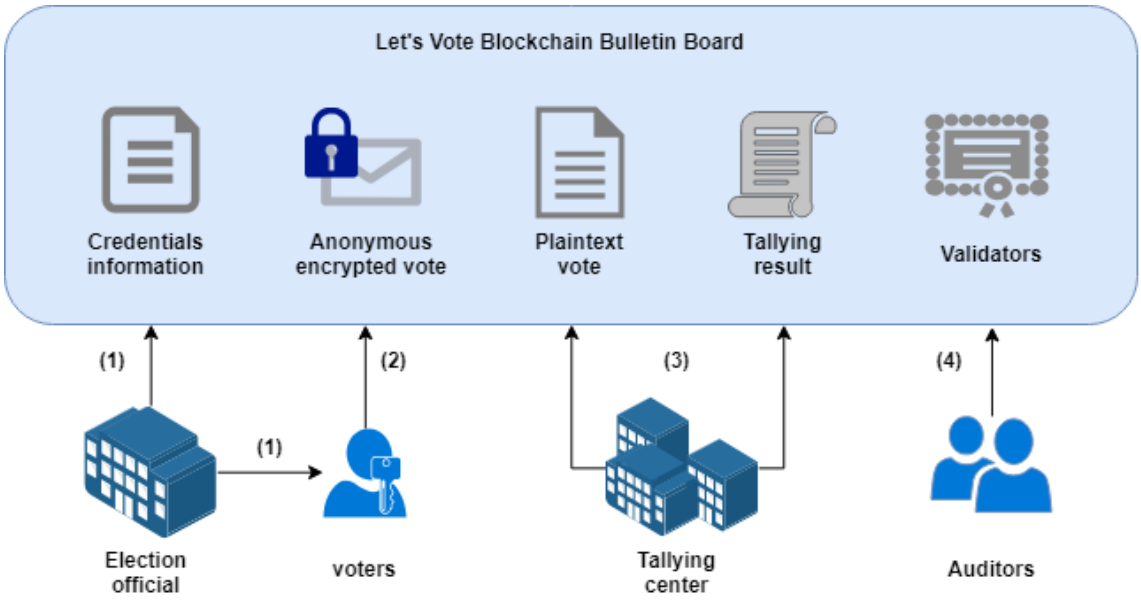

FIGURE 4.5: Voting Process. (1)Voters create their credential.Election official publish all credentials' information. (2)Voters cast their encrypted anonymous ballots. (3)Trustees decrypt the encrypted ballots, election official publish tallying result. (4)Auditors audit the result and other election processes.

- Universal verifiability. Any voter can join the election blockchain to act as an observer to monitor the whole process of the election. Because all the data generated during the election will be recorded to the blockchain, a voter could easily verify the correctness of the ballots decryption and then verify the tallying result.
- Recorded as cast: The voter could check the data on blockchain to verify that her ballot has been recorded correct or not.
- Tallied as recorded: The voter can verify the decryption information on the blockchain and tally the result by herself and then verify the official result is correct or not.

4.4 Zero-Knowledge Proof Construction

The signature of knowledge:

$$\pi_B = ZKSoK[m_\beta]\{(c, w, r) : \textbf{AccVerify}((N, u), A, c, w) = 1 \wedge c = g^S h^r\}$$

consists of the following three proofs:

1. A commitment y_1 that committed to a value(credential value) c that is accumulated in the published accumulator A.
2. A commitment y_2 is a commitment to S.
3. The two commitments y_1 and y_2 are open to the same value, the details can be found on [4].

The former proof is described in [18],we created no-interactive version of the protocol that is described below:

Suppose a voter has credential e, her witness is u ,the accumulator is v,according to the description above, they satisfy $u^e = v$. The public parameters are primes p, q such that $p = 2^w q + 1$, group generators g, h and the module $N = pq$.

The voter form a commitment to e, provided that the commitment parameters are group $\mathfrak{B}_\mathfrak{q}$, and two generator of the group are $\mathfrak{g}, \mathfrak{h}$ such that $log_\mathfrak{g}\mathfrak{h}$ is unknown. The commitment is $\mathfrak{C}_e = \mathfrak{g}^e\mathfrak{h}^r$. The following protocol is going to prove the given commitment $\mathfrak{C}_e = \mathfrak{g}^e\mathfrak{h}^r$ and a given accumulator v contain the same secret value e.

The voter need to form a commitment C_u to witness u and then prove that this commitment is the e-th root of the value v. This is carried out as follows.

1. The voter chooses $r_1, r_2, r_3 \in_R \mathbb{Z}_{[N/4]}$, calculates $C_e := g^e h^{r_1}$, $C_u := uh^{r_2}$, $C_r := g^{r_2}h^{r_3}$.

2. The voter chooses:
$$r_\alpha \in_R (-B2^{k'+k''}, ..., B2^{k'+k''})$$
$$r_\gamma, r_\varphi, r_\psi, r_\sigma, r_\xi \in_R \mathbb{Z}_q$$
$$r_\varepsilon, r_\eta, r_\zeta \in_R (-[N/4]2^{k'+k''}, ..., [N/4]2^{k'+k''}) \text{ , and}$$
$$r_\beta, r_\delta \in_R (-[N/4]q2^{k'+k''}, ..., [N/4]q2^{k'+k''})$$
Then computes:
$$t_1 := \mathfrak{g}^{r_\alpha}\mathfrak{h}^{r_\varphi}, \qquad t_2 := (\tfrac{\mathfrak{C}_e}{\mathfrak{g}})^{r_\gamma}\mathfrak{h}^{r_\psi}, \qquad t_3 := (\mathfrak{g}\mathfrak{C}_e)^{r_\sigma}\mathfrak{h}^{r_\xi},$$
$$t_1 := h^{r_\varepsilon}g^{r_\zeta}, \qquad t_2 := h^{r_\alpha}g^{r_\eta}, \qquad t_3 := C_u^{r_\alpha}(\tfrac{1}{h})^{r_\beta}, \text{ and}$$
$$t_4 := C_r^{r_\alpha}(\tfrac{1}{h})^{r_\delta}(\tfrac{1}{g})^{r_\beta}$$

3. The voter computes the hash of the message m_β(in Let's Vote, we use SHA1 as hash function)
$$c = H(m_\beta)$$

4. The voter computes
$$s_\alpha := r_\alpha - ce, \qquad s_\eta := r_\beta - cr_1, \qquad s_\varphi := r_\varphi - cr \ mod \ \mathfrak{q},$$
$$s_\beta := r_\beta - cr_2e, \qquad s_\varepsilon := r_\varepsilon - cr_2, \qquad s_\gamma = r_\gamma - c(e-1)^{-1} \ mod \ \mathfrak{q},$$
$$s_\zeta := r_\zeta - cr_3, \qquad s_\delta := r_\delta - cr_3e, \qquad s_\psi := r_\psi - cr(e-1)^{-1} \ mod \ \mathfrak{q},$$
$$s_\sigma := r_\sigma - c(e+1)^{-1} \ mod \ \mathfrak{q}, \quad \text{and} \quad s_\xi := r_\xi - cr(e+1)^{-1} \ mod \ \mathfrak{q}$$
The voter sends $t_1, t_2, t_3, t_1, t_2, t_3, t_4$ and $s_\alpha, s_\eta, s_\varphi, s_\beta, s_\varepsilon, s_\gamma, s_\zeta, s_\delta, s_\psi, s_\sigma, s_\xi$ to the verifier.

5. The verifier computes the c the same way with voter, and accept the signature if the following equations are satisfied:
$$t_1 \stackrel{?}{=} \mathfrak{C}_e^c\mathfrak{g}^{s_\alpha}\mathfrak{h}^{s_\varphi}, \qquad t_2 \stackrel{?}{=} \mathfrak{g}^c(\tfrac{\mathfrak{C}_e}{g})^{s_\gamma}\mathfrak{h}^{s_\varphi}, \qquad t_3 \stackrel{?}{=} \mathfrak{g}^c(\mathfrak{g}\mathfrak{C}_e)^{s_\sigma}\mathfrak{h}^{s_\xi}$$
$$t_1 \stackrel{?}{=} C_r^c h^{s_\varepsilon}g^{s_\zeta}, \qquad t_2 \stackrel{?}{=} C_e^c h^{s_\alpha}g^{s_\eta}, \qquad t_3 \stackrel{?}{=} v^c C_u^{s_\alpha}(\tfrac{1}{h})^{s_\beta},$$
$$t_4 \stackrel{?}{=} C_r^{s_\alpha}(\tfrac{1}{h})^{s_\delta}(\tfrac{1}{g})^{s_\beta} \qquad s_\alpha \stackrel{?}{\in} [-B2^{k'+k''}, B2^{k'+k''+1}].$$

The latter proof is detailed in [15] and it is related to the previous work [5]. In order to make it clear, we describe it as follows:

The parameters are N, g, h,the serial number S of the voter's credential $e = g^S h^r$ and a group B with its generators a, b

1. The voter makes a commitment $y = a^{g^S h^r} b^w$ to its credential $e = g^S h^r$.Let $j \leq k$ be two security parameters(we use k=160 in Let's Vote) and $H : \{0,1\}^* \leftarrow \{0,1\}^k$ be a hash function. Select $2j$ random numbers $r_1, ..., r_j$ and $v_1, ..., v_j$. for $1 \leq i \leq j$, calculate $t_i = a^{g^S h^r} b_i^v$.

2. The voter hashes the message m_β as follows.
$$c = H(m||y||g||h||a||b||S||t_1||...||t_j)$$
The zero-knowledge proof signature on the message m_β is $c, s_1, s_2, ...s_j, s'_1, s'_2, ..., s'_j$, where :
if $c[i] = 0$ **then** $s_i = r_i, s'_i = v_i$;
else $s_i = r_i - r, s'_i = v_i - wh^{r_i - r}$;

3. In order to verify the zero-knowledge proof signature, the verifier needs to compute:
$$c' = H(m||y||g||h||a||b||S||t'_1||...||t'_2)$$
where:
if $c[i] = 0$ **then** $t'_i = a^{g^S h_i^s} b^{s'_i}$;
else $t'_i = y^{h^{s_i}} b^{s'_i}$;
at the end, check whether $c = c'$

Chapter 5

Implementation of the Anonymization

5.1 Specification

The main purpose of this implementation is to demonstrate the anonymization of the Let's Vote e-voting system.The demo is based on the source code of the zerocoin.

5.2 Core Function

The core functions that are used to realize anonymity are following:

- $AccuSetup(1^{\lambda}) \rightarrow params$
- $Accumulate(params, \mathbf{C}) \rightarrow A$
- $GenWitness(params, v, C) \rightarrow w$
- $AccVerify(params, A, v, w)$
- $GeneCredential(params) \rightarrow (c, sk_c)$
- $AnonymlizeBallot(params, A, sk_c, c, w, m_\beta) \rightarrow (m_\beta, S, \pi_B)$
- $VerifyBallot(params, \pi_B, S, R, A)$

As is described in the Chapter 2, the election official are required to generate credential information (A, c, w, π) for every eligible voter. The signature of the election official π that used to authenticate the credential information is omitted, because the demo is focused on testing the performance of the anonymity.

We describe how we built and how we use these function as follows:

1. We produce n(the anonymity group size) credentials using function:
   $$GeneCredential(params) \rightarrow (c, sk_c)$$
   This function is originally from the function *MintCoin* which belongs to the class Privatecoin in the libzerocoin[8] source code. The class **Credential** in e-voting system is the same with the class **Publicoin**. The structure of the classes can be seen on figure 5.1

2. Then we compute the *accumulator* and *witness* for every credential using the functions: $Accumulate(params, \mathbf{C}) \rightarrow A$ and $GenWitness(params, v, C) \rightarrow w$

   All the credentials and corresponding accumulators and witnesses are written in the file **output**.

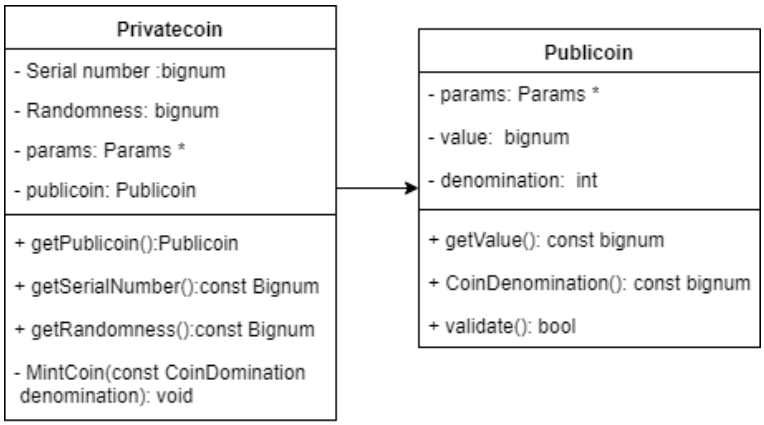

FIGURE 5.1: Class in libzerocoin[8]

3. When a voter get her information from the **output** file, she then uses the function $AccVerify(params, A, v, w)$ to verify the correctness of the *accumulator* and *witness* for her credential. If they are correct, the voter input her secret key pair (S, r) and her candidate name.

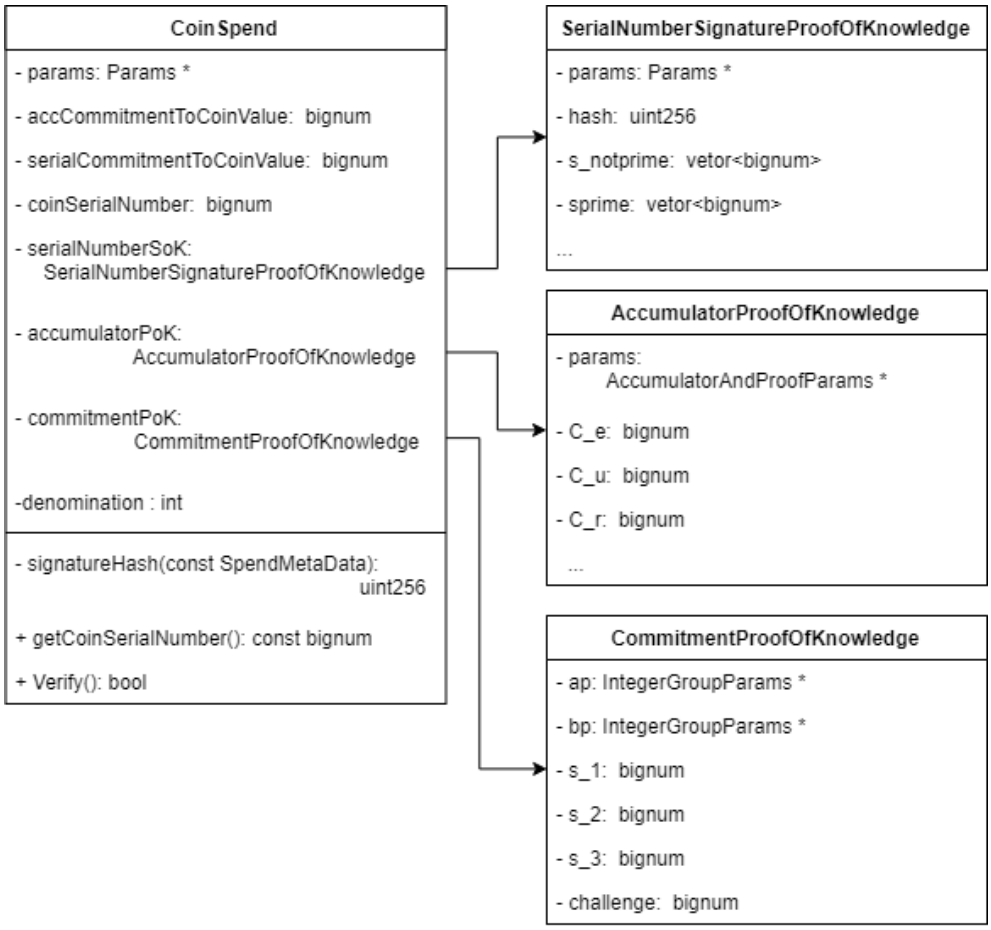

FIGURE 5.2: CoinSpend class

4. The voter uses the function $AnonymlizeBallot(params, A, sk_c, c, w, m_\beta) \rightarrow (m_\beta, S, \pi_B)$ to generate her anonymous ballot. This function is originally from the class $CoinSpend$ in the libzerocoin. The class $CoinSpend$ consists of three classes:

   - $CommitmentProofOfKnowledge$
   - $AccumulatorProofOfKnowledge$
   - $SerialNumberSignatureProofOfKnowledge$

   These three classes contain the three proofs we discuss in the chapter 4.The structure of these classes can be seen on figure 5.2

5. At the end, the voter uses the function $VerifyBallot(params, \pi_B, S, R, A)$ that belongs to the class $SpendCoin$ to check her ballot validation.

5.3 Let's vote GUI for voter

To make the voting process clear, we built a GUI based on the source code of zerocoin[8]. The GUI can be seen on figure 5.3.

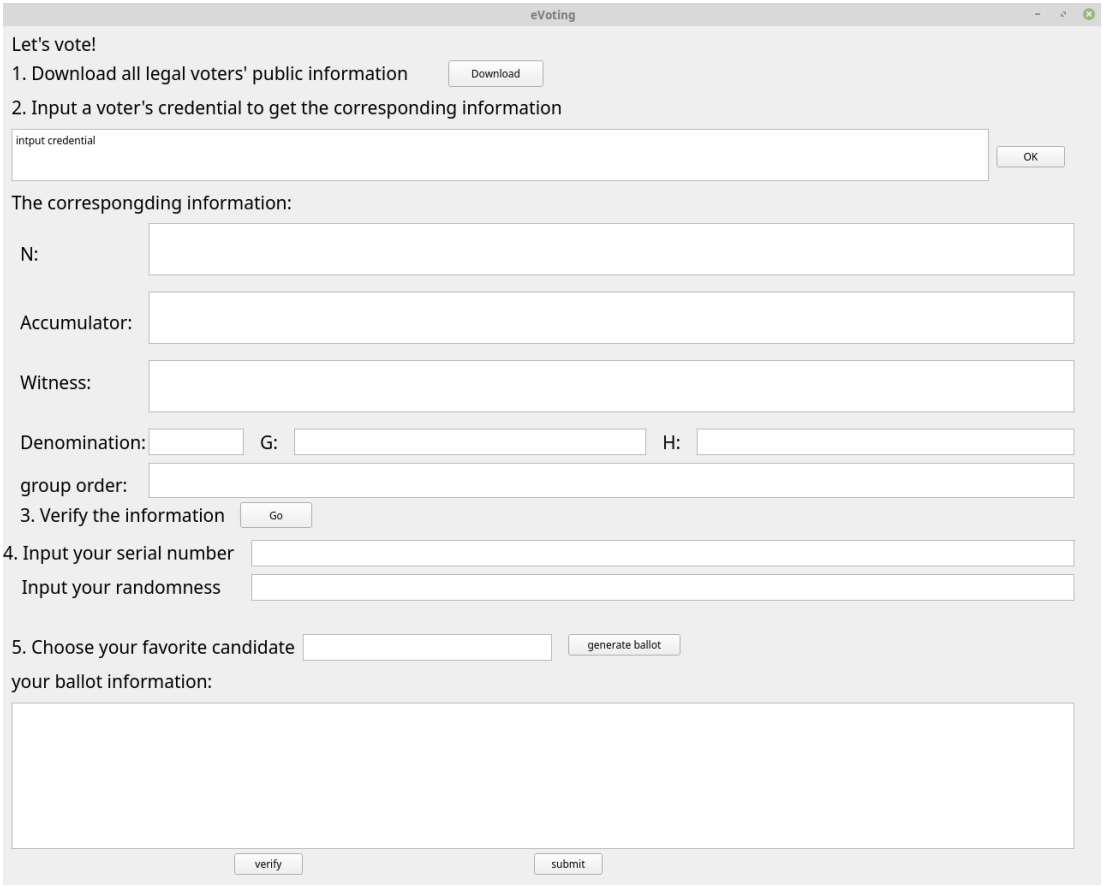

FIGURE 5.3: Voting GUI for voter

The steps to cast a ballot are as follows:
Step 1: Voter firstly inputs her credential then clicks the **OK** button, the software will download her accumulator, witness and other public *pramas*. Next she need to verify the downloaded information, this process checks if the parameters satisfy

$w^c = A$ and also checks the validation of signature that generated by the election official for these parameters. See figure 5.4

Step 2: The voter inputs her secret key pair (S, r), and the name of the candidate she chose. Then clicks the **generate ballot** button. The software will check the validation of the secret key.

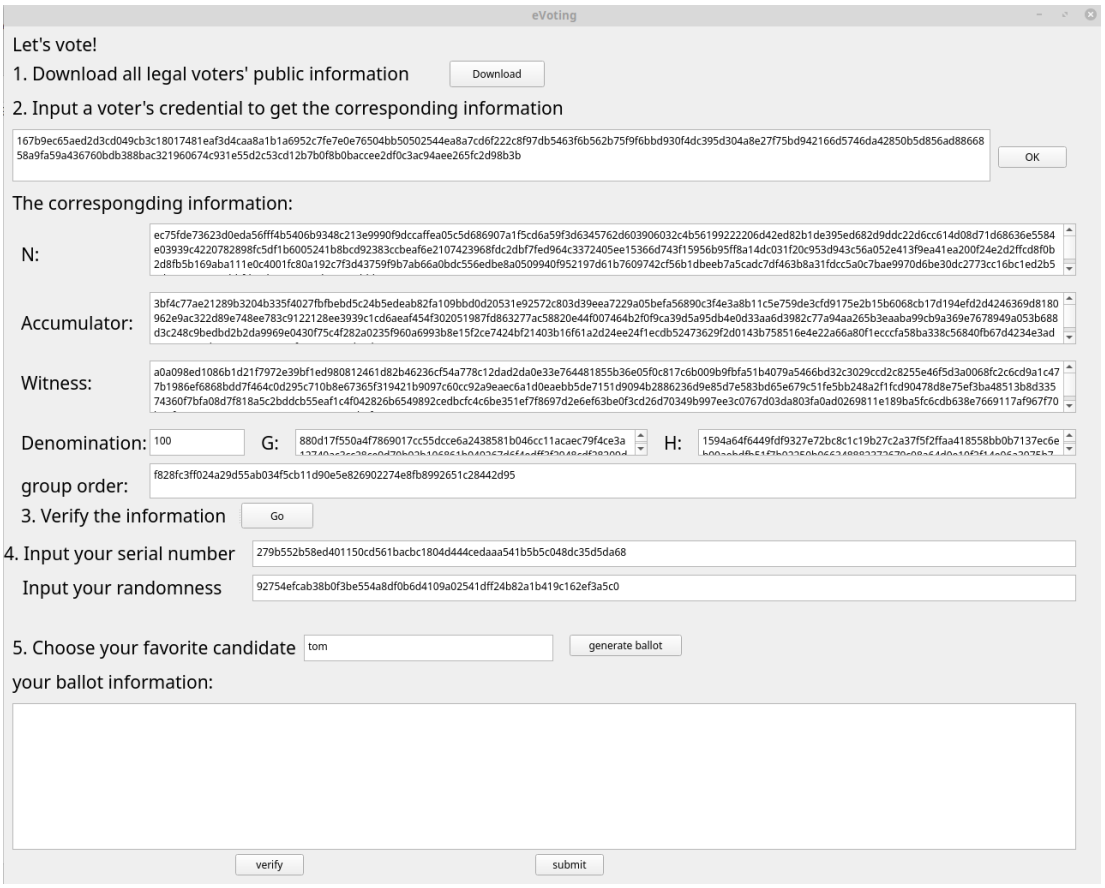

FIGURE 5.4: Voting step 2

If the secret key is valid, the software will generate ballot information and show the time to generate the zero-knowledge proof as shown in figure 5.5

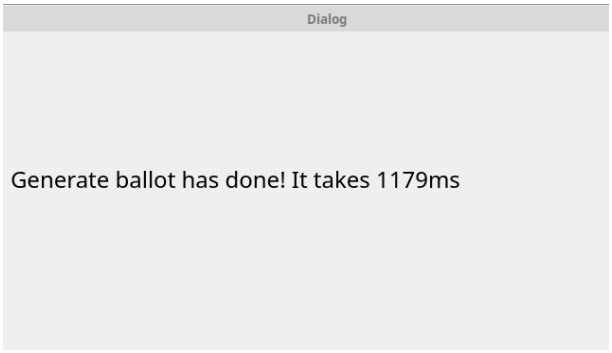

FIGURE 5.5: Voting steps 2:dialog box

Step 3: If all the information provided by the voter is valid, the voter can see her ballot information at the bottom. She can verify the validation of the ballot information in this software, she could also download other people's ballot information here to verify its validation.

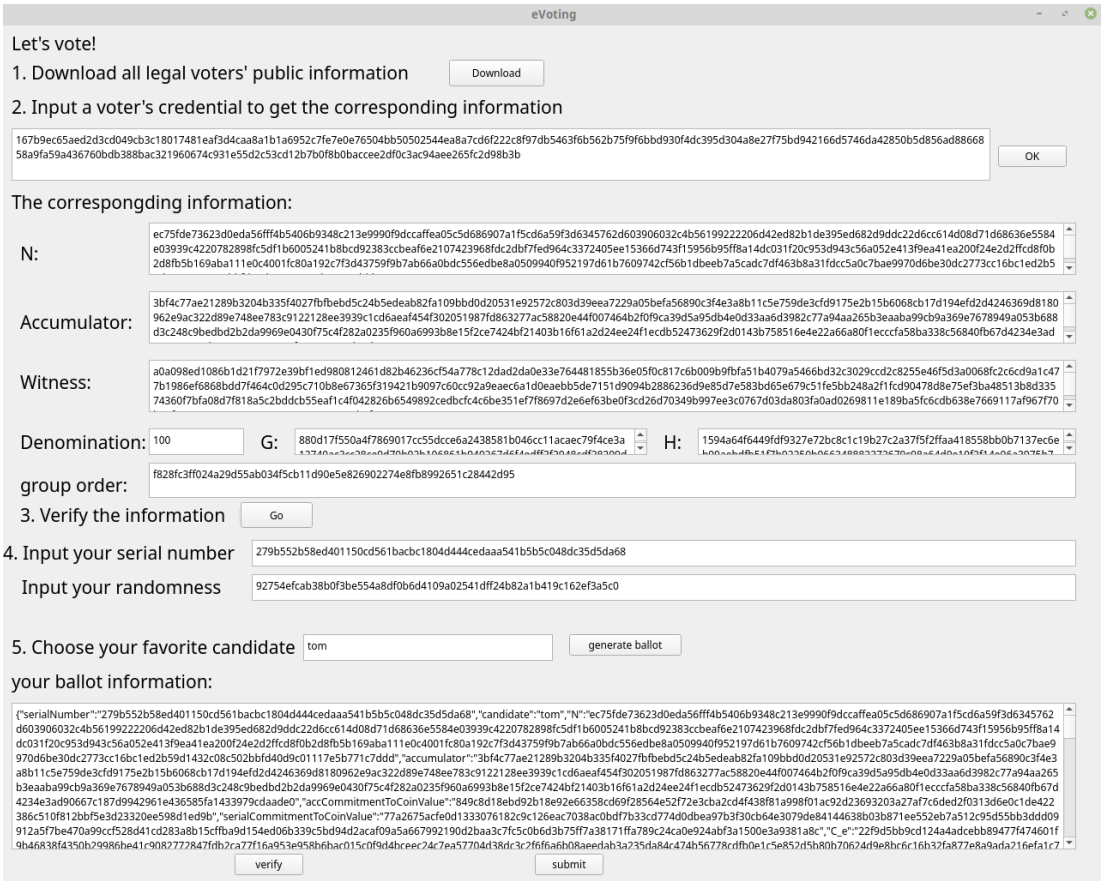

FIGURE 5.6: Voting steps 3:Ballot information on the bottom

After clicking the **Verify**, there is a dialog box to show if the ballot is valid and how long it takes to verify the ballot(zero-knowledge signature). At the end, the voter submits her ballot to the blockchain.

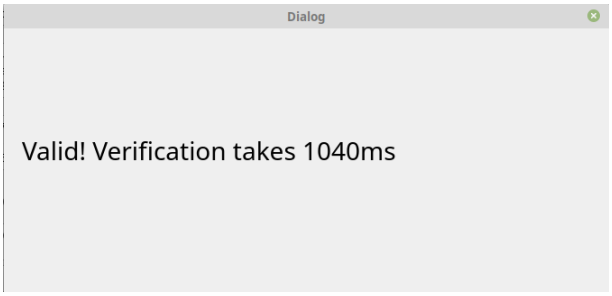

FIGURE 5.7: Voting step 3:Verification dialog

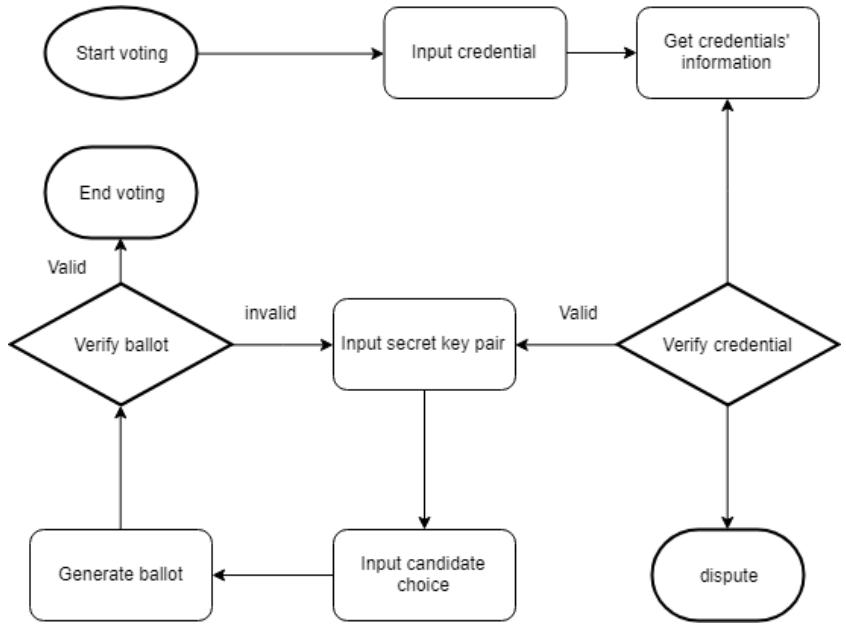

FIGURE 5.8: Voting step 3:Working flow of the GUI

5.4 Evaluation of the anonymization

We test the performance of the anonymization protocol regarding three factors: signature generation and verification time, accumulator and witness generation time, signature size. The experiments were conducted on an Intel(R) Core(TM) i5-6200U CPU(2.3GHz 2.4GHz) with 8GB of RAM, running 64-bits Windows 10.

5.4.1 Signature generation,verification time and signature size

TABLE 5.1: The signature generation, verification time and signature size

Anonymity group size	Generation time(ms)	Verify time(ms)	signature size
500	179	139	50KB
400	190	146	52.8KB
600	195	155	51.7KB
1000	186	148	53.8KB

[1] Security level parameter $\lambda = 1024$

According to the table 5.1 and figure 5.9, the time to generate and verify a signature holds steady with different anonymity group sizes(with at 50 milliseconds distinction). The same with signature size, with the same security level parameter $\lambda = 1024$, the signature size is always around 50KB. However, according to the figure 5.10 the size of the signature has a positive linear relationship with the security level parameter λ.

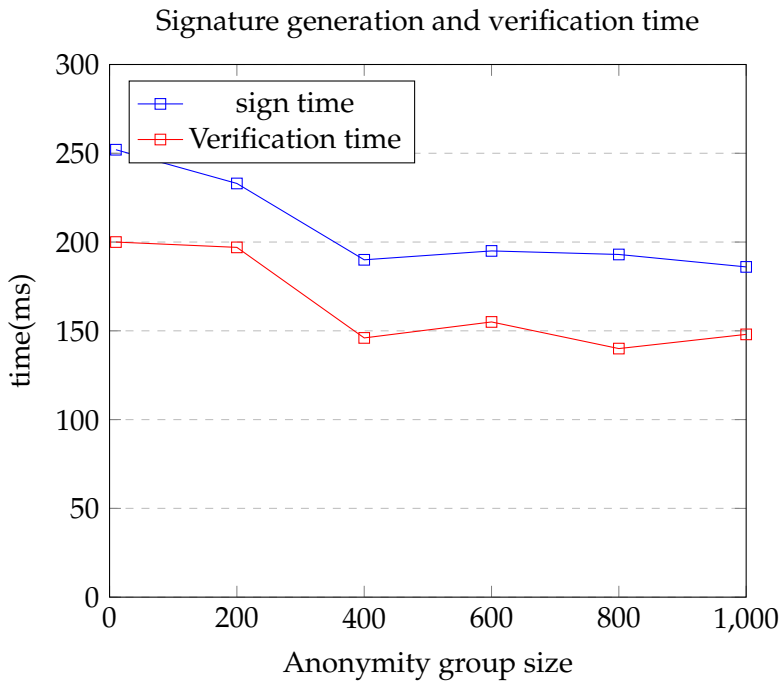

FIGURE 5.9: Signature generation and verification time

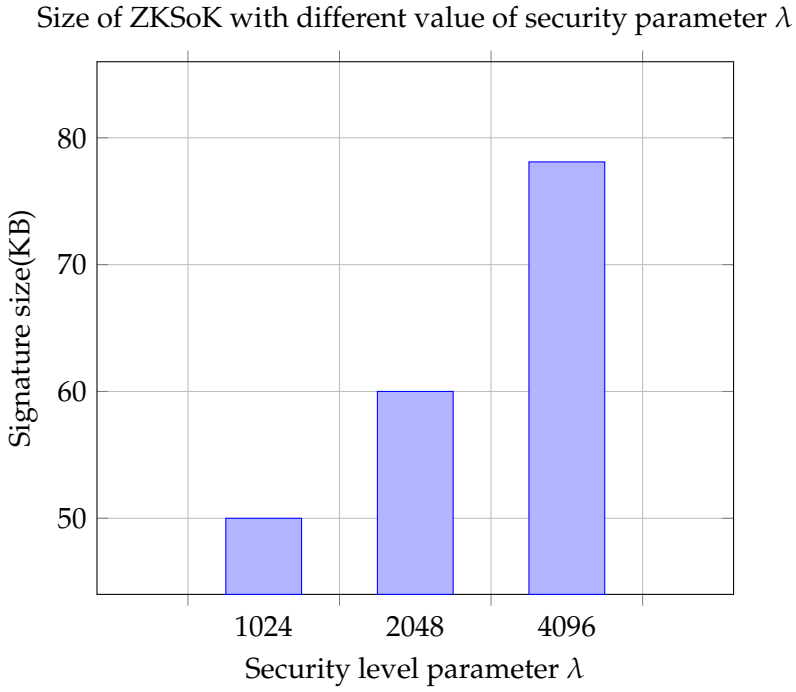

FIGURE 5.10: Size of ZKSoK with different value of security parameter λ

5.4.2 Accumulator and witness generation time

Accumulator and witness are the parameters that the election official need to generate for every voter before voting starts. According to the 5.11 the time to generate the accumulator and witness has a positive relationship with the anonymity group size. This means the more voters take part in the voting, the more time that the election official will need to do the computation. However, the computation consumption in the voter end won't change.

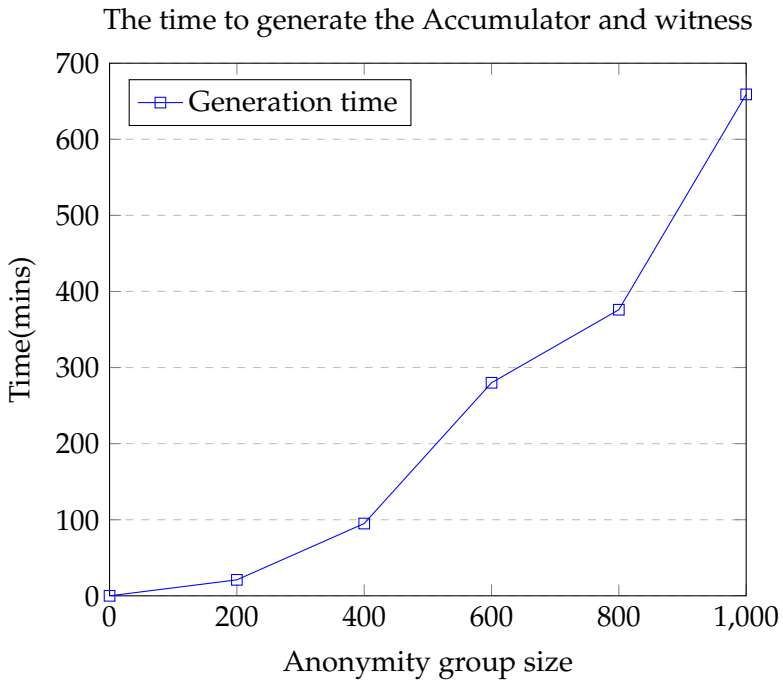

FIGURE 5.11: The time to generate the Accumulator and witness

Chapter 6

Conclusion

6.1 Conclusions

This paper evaluated the two blockchain based e-voting systems **Agora** and **Proof of Vote**.The evaluation relies on three processes: before voting, voting phase and blockchain network. We found both of these two systems use mixnet to realize anonymous ballot which is not efficient and if the mixnets nodes collude, the privacy will be broken. We created an e-voting system **Let's Vote** that uses a new method based on Zerocoin protocol to achieve the anonymity. We implemented this protocol and evaluated its performance, the result shows that the anonymization protocol of the **Let's Vote** is more efficient than the protocol based on mixnet and ring signature.

6.2 Future work

The later version of zerocoin is Zcash[9], of which the protocol is much more complex than that of Zerocoin. But the Zcash is safer and more efficient. In the future, it is worthwhile to improve Let's Vote system by the inspiration from Zcash.

Let's Vote, **Agora** and **Proof of vote**, none of them achieve receipt freeness because they all use ElGamal encryption. If the voter gives the random number r, she can prove how she voted. In the future, we should find a better way to encrypt the ballot to achieve better ballot privacy.

Appendix A

A.1 Git project links:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2017/sxw747.git

A.2 Git project content:

- e-voting: The **Let's Vote** e-voting Qt GUI for the voters.
- libzerocoin: The executable file that the GUI needs to run.

Bibliography

[1] Man Ho Au et al. "Short Linkable Ring Signatures Revisited". In: *Public Key Infrastructure*. Ed. by Andrea S. Atzeni and Antonio Lioy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 101–115. ISBN: 978-3-540-35152-8.

[2] *Simple Verifiable Elections*. 2006. URL: https://www.microsoft.com/en-us/research/publication/simple-verifiable-elections/.

[3] "Bitcoin Developer Guide". In: (). URL: https://bitcoin.org/en/developer-guide#block-chain.

[4] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 431–444. ISBN: 978-3-540-45539-4.

[5] Jan Camenisch. "Group signature schemes and payment systems based on the discrete logarithm problem". In: 1998.

[6] Jan Camenisch and Anna Lysyanskaya. "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials". In: *Advances in Cryptology — CRYPTO 2002*. Ed. by Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 61–76. ISBN: 978-3-540-45708-4.

[7] David Chaum. "Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA". In: (1988). Ed. by D. Barstow et al., pp. 177–182.

[8] Ian Miers ChristinaL Garman. "libzerocoin git hub". In: ().

[9] Christina Garman Matthew Green Ian Miers Eran Tromer Madars Virza Eli Ben-Sasson Alessandro Chiesa. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: (2014). URL: https://z.cash/technology/.

[10] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: (1987). Ed. by Andrew M. Odlyzko, pp. 186–194.

[11] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. "A practical secret voting scheme for large scale elections". In: *Advances in Cryptology — AUSCRYPT '92* (1993), pp. 244–251.

[12] Adem Efe Gencer, Robbert van Renesse, and Emin Gün Sirer. "Short Paper: Service-Oriented Sharding for Blockchains". In: *Financial Cryptography and Data Security*. Ed. by Aggelos Kiayias. Cham: Springer International Publishing, 2017, pp. 393–401. ISBN: 978-3-319-70972-7.

[13] Essam M. Ghadafi. "Sub-linear Blind Ring Signatures without Random Oracles". In: *Cryptography and Coding*. Ed. by Martijn Stam. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 304–323. ISBN: 978-3-642-45239-0.

[14] Jens Groth. "A Verifiable Secret Shuffle of Homomorphic Encryptions". In: *Journal of Cryptology* 23.4 (2010), pp. 546–579. ISSN: 1432-1378. DOI: 10.1007/s00145-010-9067-9. URL: https://doi.org/10.1007/s00145-010-9067-9.

[15] Matthew Green Aviel D. Rubin Ian Miers Christina Garman. "Zerocoin: Anonymous Distributed E-Cash from Bitcoin". In: *2013 IEEE Symposium on Security and Privacy* ().

[16] "Internet voting: coming to a computer near you, though more research is needed to eliminate the risks." In: (2014).

[17] Markus Jakobsson. "A practical mix". In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 448–461. ISBN: 978-3-540-69795-4.

[18] Anna Lysyanskaya Jan Camenisch. "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials". In: (2002), pp. 61–76.

[19] Byoungcheon Lee et al. "Providing Receipt-Freeness in Mixnet-Based Voting Protocols". In: *Information Security and Cryptology - ICISC 2003*. Ed. by Jong-In Lim and Dong-Hoon Lee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 245–258. ISBN: 978-3-540-24691-6.

[20] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups". In: *Information Security and Privacy*. Ed. by Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 325–335. ISBN: 978-3-540-27800-9.

[21] Joseph K. Liu and Duncan S. Wong. "Linkable Ring Signatures: Security Models and New Schemes". In: *Computational Science and Its Applications – ICCSA 2005*. Ed. by Osvaldo Gervasi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 614–623. ISBN: 978-3-540-32044-9.

[22] Andrew Miller et al. "An Empirical Analysis of Linkability in the Monero Blockchain". In: *CoRR* abs/1704.04299 (2017). arXiv: 1704.04299. URL: http://arxiv.org/abs/1704.04299.

[23] Kirill Nikitin et al. "CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1271–1287. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/nikitin.

[24] Miyako Ohkubo et al. "An Improvement on a Practical Secret Voting Scheme". In: *Information Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 225–234. ISBN: 978-3-540-47790-7.

[25] Tatsuaki Okamoto. "Receipt-free electronic voting schemes for large scale elections". In: *Security Protocols*. Ed. by Bruce Christianson et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 25–35. ISBN: 978-3-540-69688-9.

[26] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. "Efficient Anonymous Channel and All/Nothing Election Scheme". In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 248–259. ISBN: 978-3-540-48285-7.

[27] Torben Pryds Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology — CRYPTO '91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140. ISBN: 978-3-540-46766-3.

[28] Kun Peng, Colin Boyd, and Ed Dawson. "Simple and Efficient Shuffling with Provable Correctness and ZK Privacy". In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 188–204. ISBN: 978-3-540-31870-5.

[29] Feng Hao Peter Y. A. Ryan. "Real-World Electronic Voting Design, Analysis and Deployment". In: (2016).

[30] Gareth William Peters and Efstathios Panayi. "Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money". In: *CoRR* abs/1511.05740 (2015). arXiv: 1511.05740. URL: http://arxiv.org/abs/1511.05740.

[31] "PoA network white paper". In: (). URL: https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper.

[32] "Proof of Vote White Paper". In: (2018).

[33] D. Wong S. S. M. Chow J. Liu. "Robust Receipt-Free Election System with Ballot Secrecy and Verifiability". In: *NDSS Symposium 2008* (2008).

[34] Kazue Sako and Joe Kilian. "Receipt-Free Mix-Type Voting Scheme". In: *Advances in Cryptology — EUROCRYPT '95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 393–403. ISBN: 978-3-540-49264-1.

[35] Hovav Shacham and Brent Waters. "Efficient Ring Signatures Without Random Oracles". In: *Public Key Cryptography – PKC 2007*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 166–180. ISBN: 978-3-540-71677-8.

[36] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: http://doi.acm.org/10.1145/359168.359176.

[37] Sorrge. "LSAG". In: (2013). URL: https://github.com/sorrge/LSAG.

[38] Travis Springall Drew Finkenauer et al. "Security Analysis of the Estonian Internet Voting System". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14 (2014), pp. 703–715. URL: http://doi.acm.org/10.1145/2660267.2660315.

[39] Shi-Feng Sun et al. "RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero". In: (2017). Ed. by Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, pp. 456–474.

[40] Agora team. "Agora while paper version0.2". In: (2018). URL: https://www.agora.vote/.

[41] Viktoria Ronge Dominique Schröder Tim Ruffing Sri Aravinda Thyagarajan. "Burning Zerocoins for Fun and for Profit". In: (1991). URL: https://zcoin.io/statement-paper-burning-zerocoins-fun-profit/.

[42] A. Tomescu and S. Devadas. "Catena: Efficient Non-equivocation via Bitcoin". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 393–409. DOI: 10.1109/SP.2017.19.

[43] "Tor project,privacy online". In: (). URL: https://www.torproject.org/.

[44] Patrick P. Tsang and Victor K. Wei. "Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation". In: *Information Security Practice and Experience*. Ed. by Robert H. Deng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 48–60. ISBN: 978-3-540-31979-5.

[45] Dong Zheng et al. "Linkable Ring Signatures from Linear Feedback Shift Register". In: *Emerging Directions in Embedded and Ubiquitous Computing*. Ed. by Mieso K. Denko et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 716–727. ISBN: 978-3-540-77090-9.