Publicity Verifiable Ranked Choice Online Voting System

A thesis submitted in fulfilment of the requirements
for the degree of DOCTOR OF PHILOSOPHY.

XUECHAO YANG

Bachelor of Information Technology (with Distinction) RMIT University;
Bachelor of Computer Science (Honours 1st Class) RMIT University.

School of Science
College of Science, Engineering and Health
RMIT UNIVERSITY

OCTOBER, 2018

Author's declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

SIGNED: XUECHAO YANG ..... DATE: 22 October 2018 .....

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my senior supervisor Prof. Xun Yi for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I would also like to thank my associate supervisors Dr. Fengling Han and Dr. Surya Nepal, who guided me to accomplish my research goals. Her endless encouragement and continuous support has always kept me motivated and ensured a smooth candidacy for completion of this research. I could not have imagined having a better supervision team for my Ph.D. study, which helped me reach the finishing line of my PhD journey in RMIT.

I appreciate the support of School of Science RMIT and Data61CSIRO for providing scholarship and a good research environment. I am also thankful to my colleague and mentor, Dr. Andrei Kelarev, who helped me a lot with suggestions and advice during my research.

I would also like to thank my family. I am grateful for the sacrifice they made for me. Especially to my partner Miss Meng Meng, who always provided me mental strength and courage while being thousands of kilometres away.

Last but not least, I would like to express my gratitude to all my friends in and outside of RMIT that directly or indirectly supported me throughout this time. I hope my research findings would benefit the research community.

Publications

- X. Yang, X. Yi, C. Ryan, R. V. Schyndel, F. Han, S. Nepal, and A. Song. “A verifiable ranked choice internet voting system”. In International Conference on Web Information Systems Engineering, 490-501, 2017. **(CORE Rank A) - Chapter 3.**

- X. Yang, X. Yi, S. Nepal, A. Kelarev and F. Han. “A Secure Verifiable Ranked Choice Online Voting System Based on Homomorphic Encryption”. IEEE Access, 6, 20506-20519, 2018. **(SJR Q1) - Chapter 4.**

- X. Yang, X. Yi, S. Nepal, and F. Han. “Decentralized Voting: A Self-tallying Voting System Using a Smart Contract on the Ethereum Blockchain”. In International Conference on Web Information Systems Engineering, 2018. **(CORE Rank A - Accepted to publish) - Chapter 5.**

- X. Yang, X. Yi, S. Nepal, A. Kelarev and F. Han. “Blockchain Voting: Decentralised Publicly Verifiable Online Voting Protocols”. Future Generation Computer Systems, 2018. **(SJR Q1 - Under second round review) - Chapter 6.**

Abstract

Elections conducted on paper consume a lot of resources and contribute to the destruction of forests, which leads to climate deterioration. Moreover, such election process can make it difficult for some people to vote and it often leads to doubts in the validity of counting, in people submitting multiple votes, in ineligible people voting. In several well-known previous examples, doubts in the validity of paper elections lead to the need of recounting and even court battles to decide the validity of the outcome. Having a way to vote online could be an easier and more reliable solution. However, secure and verifiable methods of online voting need to be developed to achieve this.

Recent online voting experiences in countries such as the United States, India and Brazil demonstrated that further research is needed to improve security guarantees for future elections, to ensure the confidentiality of votes and enable the verification of their integrity and validity. Electronic voting, to be successful, requires a more transparent and secure approach, than the approach that is offered by current electronic voting protocols. Advanced security methods are necessary to introduce effective online voting in the whole world.

Currently, most online voting systems are centralized, which means that they involve central tallying authorities to take responsibility for verifying, tallying and publishing the final outcome of the election. These previous systems always assume that their central authorities are honest. Otherwise, the published final outcome cannot be trusted. The aim of our new research is to propose and investigate a decentralized ranked choice online voting systems, which never rely on any third party (such as tallying authorities), thereby significantly increasing the confidence and

trust of the voters.

The thesis presents several publicly verifiable online voting systems and indicates the processing steps and stages in the development of a publicity verifiable online voting system from centralized to semi-decentralized, to fully decentralized. By using Homomorphic cryptosystem, proof of zero knowledge and Blockchain technology, the proposed system in this thesis can achieve the following: (1) Flexible voting mechanism: voters can easily rank all candidates; (2) Publicity verifiable: the whole election procedure is transparent and verifiable by voters; (3) Self-tallying: the final outcome of the election can be computed by any individual voter; and (4) Fully decentralized: no tallying authority (or any other trusted third party) involved at all.

The proposed systems presented in this thesis include protocols developed on Blockchain technology. The particular technology that is used as the basis for a secure online voting system is “smart contract over Blockchain”, which offers a factor of the integrity of votes and has not been deeply studied in Blockchain technologies to date. The proposed voting protocols ensure confidentiality and preserve the voters’ privacy while keeping the election procedures transparent and secure. The underlying Blockchain protocol has not been modified in any way, the voting scheme proposed merely offers an alternative use case of the protocol at hand, which could be presented as the basis for voting systems using Blockchain with further development of the underlying Blockchain protocols.

Table of Contents

List of Tables

List of Figures

Abbreviations and notations

V_i	i-th Voter; $i \in [1, n_v]$
n_v	number of Voters
pk_{v_i}	public key of v_i
sk_{v_i}	secret key of v_i
B_i	the ballot submitted by V_i; $i \in [1, n_v]$
Sig_{v_i}	digital signature of v_i
C_j	j-th Candidate; $j \in [1, n_c]$
n_c	number of Candidates
pk_{c_j}	public key of c_j
sk_{c_j}	secret key of c_j
A_i	i-th Authority; $i \in [1, n_a]$
n_a	number of Authorities
pk_{a_i}	public key of a_i
sk_{a_i}	secret key of a_i
PK	common public key for encrypting ballots
PZK{...}	proof of zero knowledge
PPK{...}	proof of partial knowledge

PVRC	Public Verifiable Ranked Choice Online Voting System
LSPVRC	Large-Scale Public Verifiable Ranked Choice Online Voting System
SDPVRC	Semi-Decentralized Public Verifiable Ranked Choice Online Voting System
STDPVRC	Self-Tally Decentralized Public Verifiable Ranked Choice Online Voting System

Chapter 1

Introduction

1.1 Background and Motivation

Traditional voting systems require each voter to cast a ballot with pen and paper. In 2000 United States presidents election, George W. Bush and Al Gore participated in the contest. The result of the election hinged on Florida outcome, and the votes awarded it to Bush. However, the result was close enough to perform an additional recount under state law, and Gore believed he could win after undervotes [AP02] were tallied. Bush did not agree with the recount, and the litigation reached the U.S. Supreme Court. Eventually, the court decided to end the recount thereby effectively granting Bush the victory. Nevertheless, the controversy did not end as well, a lot of people thought Gore could win if the U.S. Supreme Court did not stop the recount [Ket01, Par07, Cha12, WPD15]. There are several potential problems that could occur in paper based ballots, such as missed ballots and unrecognised ballots.

Elections conducted on paper consume a lot of resources and contribute to the destruction of forests, which leads to climate deterioration. In recent years, online voting (which can also be called electronic voting, e-voting, Internet voting) systems have allowed voters to cast their votes electronically from Internet connected devices. Such systems reduce the cost of an election and increase voters' participation (especially for voters with disabilities) due to the convenient voting

procedure [ALPL13].

One of the most obvious, direct ways to influence a country's election is to interfere with the way citizens actually cast votes. As the United States (and other nations) embrace electronic voting, it must take steps to ensure the trustworthiness of the systems. Not doing so can endanger a nation's domestic democratic will and create general political discord — a situation that can be exploited by an adversary for its own purposes [Wil16].

The process of online voting (such as democratic voting) requires a strong sense of trust, otherwise, misleading maps and distorted data from any process of the election can easily lead to an inaccurate rumour. For example, there was a news saying that "Donald Trump won 3084 of America's 3141 counties in the 2016 presidential election; Hillary Clinton won just 57." [MOR16]. Obviously, that was not true, as the article offered no citations or links to support its claims [LaC16]. However, there is also no evidence to show that the news was fake, because the processing procedure of the system was not transparent to the public, and even the voters did not understand how the election outcome was computed [Bea16, Rob16, Bre16].

Thus, advanced security methods are necessary to introduce effective online voting in the whole world. Recent online voting experiences in countries such as the United States, India and Brazil demonstrated that further research is needed to improve security guarantees for future elections, to ensure the confidentiality of votes, enable the verification of their integrity and validity, and most importantly, the transparency of the whole election processing procedure to build the trust in the system.

1.1.1 Ranked Choice Voting Mechanism

Online voting systems have been used for government elections in many countries, such as United States, United Kingdom, France, Canada etc. However, most of current voting systems only used plurality voting [MPRJ10], but not preferential voting [Top10].

Plurality voting, also known as first-past-the-post, has been most widely used throughout

the world. Under the plurality voting system, each voter can only choose one candidate from a list of candidates, such as (a) of Figure 1.1. The candidate who received the highest number of votes wins the election. However, this system is only suited to the two-candidate election. If there are multiple (more than 2) candidates, and two of them are predicted to win the election. Under this situation, voters are pressured to vote for one of popular candidates rather than their true preference. The votes for any other candidate are likely wasted because those votes might have no impact on the final result.

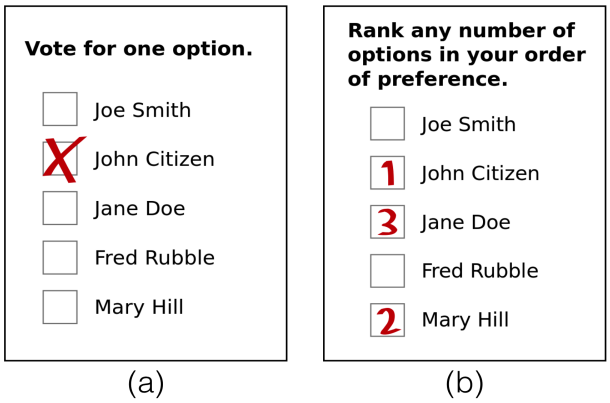

Figure 1.1: (a) cast ballot under plurality voting mechanism; (b) cast ballot under preferential voting mechanism.

Preferential voting, also known as ranked choice voting, is a voting system used to elect a winner from multiple candidates. Under the preferential voting system, voters are allowed to rank all candidates according to their personal preferences rather than voting for only a single candidate, such as (b) of Figure 1.1. According to the different places assigned to the candidates in the ranked list, the candidates received different points from each ballot.

1.1.2 Security Requirements

Electronic voting protocols have been around for a while. They have been implemented in different elections, ranging from university to government based elections. Maintaining the privacy and security of the voters is a priority for any online voting system. Our voting system ensures that the following security requirements are met. These requirements are essential for voting

according to [DLL12, SCM08]. They are shown in Table 1.1.

Table 1.1: Security requirements of an online voting system.

Eligibility of voters:	Only authorized voters can submit votes.
Multiple-voting detection:	Each voter can only vote once. Multiple voting by any one voter is detected and identified.
Privacy of voters:	All votes must be stored securely and secretly and should not reveal voting preferences of the voters.
Integrity of vote:	No one can modify or duplicate any submitted vote without being detected.
Correctness of tallied result:	Only verified votes are counted and added to the final result.
End-to-End Voter Verifiable:	Every voter is able to verify whether their vote is posted and counted correctly.

The process of online voting (such as democratic voting) requires a strong sense of trust — in the equipment, the process and the people involved. Many viable protocols have been created since Chaum [Cha04] first proposed Votegrity, which is one of the first end-to-end (E2E) verifiable voting solutions. An E2E voter verifiable voting system should achieve the following:

- Any voter is able to verify that his/her own vote has been cast as intended based on the receipt provided by the electronic voting booth, and by verifying that their vote has propagated to the public web bulletin board correctly.

- Any voter is able to verify that his/her vote has been counted correctly and included in the final tally outcome.

- Any voter is able to verify that who may not be involved in the election or voting, in order to ensure that no votes have not been compromised.

Some of the most prominent examples that have stemmed from Chaum's Votegrity are Markpledge [Nef04], Prêtà voter [RBH+09], Helios [Adi08], Scantegrity [CCC+10] and STAR-Vote [BBB+13]. These systems also provide E2E verifiability. Helios 2.0 [ADMP+09], Apollo [GKV+16] and Zeus [TPLT13] were developed based on Helios.

All these centralized online voting systems use a public web bulletin board (WBB) for posing all of the cast ballots for the public to see. Web bulletin boards in these systems are used as an authenticated public broadcast channel which, as mentioned above, displays the cast ballots to the public in an encrypted form. The WBBs serve as an important stage for any E2E protocol. Typically, after the voter has cast their vote and received a receipt encrypting their choice in a way that is dependent of what voting protocol used, the encrypted vote is propagated to the WBB.

The receipt is an important part of the voting protocol, as it allows the user to prove their vote to an authority in case the voter wishes to dispute their vote or prove that they have voted contrary to what the system has recorded. The receipt also allows the user to find his/her vote and view how the system recorded the vote. These receipts vary from system to system and so does the way the voter verifies these receipts. Typically these receipts contain a summary of how the voter voted, which can be presented to the voter in an encrypted or obfuscated manner.

However, for the previous centralized systems, the security level depends on the centralized server and authorities. For example, if the content is posted on a WBB and the system provides receipts, then using such a system, we have to assume that the authorities never collude and that the centralized server cannot be compromised.

1.2 Our Research Questions

From the investigation of the protocols, it follows that an electronic voting system must be secure while allowing for as much transparency as possible to be a working E2E verifiable. In order to enhance the confidentiality of voters, the decentralized online voting system is also proposed and considered, where the dependence of authorities is reduced.

Blockchain [Nak08] helps to achieve this level of security and transparency, while maintaining privacy and non-malleability of the transactions, which may indeed be the future of online voting protocols. The benefits of using Blockchain are in its decentralised nature, relatively low cost of transactions and tamper-proof properties, which play an important role if a voting system is

based on this technology [Del17, Wes16].

By using Blockchain technology, a decentralized publicity verifiable online voting system is expected to be developed, which should achieve advanced E2E voter verifiable as below:

- Any voter is able to verify that his/her own vote which has been cast as intended and stored correctly without any receipt that is provided by the centralized system.
- Any voter is able to verify the correctness of the final tally outcome, and even able to compute final tally outcome individually.
- Since all votes are stored in the Blockchain database (instead of WBB in centralized systems), no one is able to compromise/modify any confirmed submission. Any voter is able to verify the correctness/eligibility of any other vote in the Blockchain.

In order to propose a ranked choice online voting system that achieved the advanced E2E verifiability, the following questions guide the research:

1. What is the strategy to construct each ballot with preferential ranking information without revealing any voter's privacy but making it individually verifiable?
2. How to achieve a large-scale verifiable ranked choice online voting system with verifiable processing procedure and practical performance?
3. How to reduce the importance and dependence of the tallying authorities for a public verifiable online voting system by using Blockchain technology?
4. How to achieve a Blockchain-based decentralized online voting system with individual and universal verifiability and self-tallying by any voter, but without any tallying authority?

1.3 Aims and Objectives

We aim to propose an online voting system, which not only allows voters to rank all candidate rather than vote for one, and the outcome of the election is made publicly verifiable, but also we aim to enhance the trust of the system and the confidence in the votes by reducing the dependence on tallying authorities, and even proposing a system without any tallying authority. The primary aims of this thesis are:

- Design and propose different suitable voting mechanisms for online voting systems, which will not only allow voters to rank all candidates of an election easily, but will also keep all submissions of the voters secret all the time.
- Design and propose more practical/feasible online voting systems with larger scale of voters, without affecting the usability of the voting mechanism and the privacy of the voters.
- Design and propose decentralized online voting systems which can reduce the importance and the dependence of tallying authorities without affecting the performance and security level.
- Design and propose a self-tallying decentralized online voting system without any tallying authority, but fulfilling all security requirements with a reasonable performance.

1.4 Contributions of this thesis

All our proposed online voting systems achieve the following. Authorized voters are able to register via Internet by submitting their identification documents (e.g. passport or driver's licence). Therefore, no one can register more than once, and double voting can also be prevented, because the identities of voters are public values. Each submitted ballot is encrypted by a probabilistic encryption, meaning that the ciphertexts are always different every time even if the inputs are the same. Thus, the ciphertexts can be treated as a proof (receipt) of any particular

submission, and used to verify whether a submitted ballot has been posted correctly into the pool. Everyone is able to verify the eligibility of any voter's ballot without revealing voter's privacy. Once all voters submitted their ballots, the voting information of each encrypted ballot (only verified ballots) can be added to other ballots directly, and the correctness of the final tallied outcome is verifiable. Furthermore, any compromised authority can be easily detected by authorities or voters because the authorities are required to post every step of their work (based on proof of zero knowledge) into the public bulletin board.

To sum up, we proposed online voting systems which not only satisfy requirements, but also reduce the importance and dependence of the tallying authorities in the system from building a centralized system to a decentralized system by using Blockchain technology. Furthermore, our proposed self-tallying protocol allows the tallying procedure of the election to be processed by only all voters and candidates, without a need of any tallying authority to get involved. The contributions of this thesis as the following.

- To the best of our knowledge, our proposed system is the first decentralized ranked choice online voting system in existence, where voters are able to cast their ballots by assigning different scores to different candidates (ranking all candidates), rather than only voting for one candidate.

- We propose a new encryption mechanism, which merges two previously known approaches and combines the use of the secret keys of all voters and the secret keys of all candidates of the election.

- We propose a new implicit verification algorithm to verify the validity of all votes submitted to the election without revealing any voter's privacy.

- Our proposed protocols ensure security, privacy and public verifiability of the whole election including the tallying and vote verification processes, which do not rely on a trusted tallying authority.

- No adaptive issue in our proposed self-tallying protocol. In our proposed decentralized system, the individual who votes last cannot tally the result earlier than other voters.
- The importance and dependence of the tallying authority are reduced, and he/she can never reveal any voter's privacy. Even there is no tallying authority needed at all.

1.5 Structure of this thesis

This thesis is organized as follows:

- **Chapter 2:** briefly introduces the preliminaries on the work about online voting systems. There are 3 parts in the chapter: 1) we provide the underlying cryptographic algorithms for all of our proposed systems, such as ElGamal cryptosystem; 2) we describe how the homomorphic addition is applied to the online voting systems; 3) the background of decentralized online system is also presented.
- **Chapter 3:** A **PVRC** (public verifiable ranked choice) online voting system is proposed, which allows each voter to rank all candidates of the election rather than vote-for-one. And all encrypted ballots are kept as secret all the time after submission, which can be tallied directly without accessing any submitted ballot.
- **Chapter 4:** A **LSPVRC** (large-scale public verifiable ranked choice) online voting system is proposed, where the voting mechanism is upgraded in comparison to the **PVRC** system, in order to improve the computation performance of the system, where it allows each voter to score different candidates of the election, but the voting preference will not be revealed after ballot submission.
- **Chapter 5:** A **SDPVRC** (semi-decentralized public verifiable ranked choice) online voting system is proposed by using Blockchain technology, in order to reduce the dependence on the tallying authorities. According to the voting mechanism in **LSPVRC** system, we

re-designed the proof generation and verification protocols, which means that only one tally authority is needed, and he/she can never reveal any voter's submission.

- **Chapter 6:** A **STDPVRC** (self-tally decentralized public verifiable ranked choice) online voting system is proposed, which is a self-tallying system without any tallying authority. In order to enhance the vote confidentiality of the system, the voting mechanism is re-designed (inspired by the **LSPVRC** and **SDPVRC**), and a self-tallying protocol is proposed, which can be used by any voter individually.

- **Chapter 7:** concludes this thesis.

Chapter 2

Literature Review

In this section, we introduce prerequisites for the underlying cryptographic algorithms, which are used as building blocks in our proposed online voting systems.

In the rest of this chapter, we describe the related works of Homomorphic based online voting systems and Blockchain based online voting systems.

2.1 Underlying Cryptographic Algorithm

2.1.1 ElGamal Cryptosystem

ElGamal cryptosystem is very well known (cf. [AARTAD17] and [YPB14]). We assume that the cyclic group (G, q, g), secret key x and public key $y = g^x$ are defined.

Encryption: To encrypt a plaintext message $m \in G$:

- Randomly choose an integer r from $\mathbb{Z}_q^*$;
- Computes $c_1 = g^r$;
- Computes $c_2 = g^m \cdot y^r$.

The encrypted message is $E(m) = (c_1, c_2)$.

Decryption: To decrypt a ciphertext message (c_1, c_2):

- computes a partial decryption value ${c_1}^x$
- computes $\frac{c_2}{{c_1}^x} = \frac{g^m \cdot y^r}{g^{r \cdot x}} = \frac{g^m \cdot g^{x \cdot r}}{g^{r \cdot x}} = g^m$.

Finally, m can be revealed by computing a discrete logarithm.

2.1.1.1 Distributed ElGamal Cryptosystem

We assume there are n users in the system. Each i-th user has its own public key y_i and secret key x_i. The distributed ElGamal cryptosystem consists of the following algorithms.

Key generation: A common public key

$$PK = \prod_{i=1}^{n} y_i = g^{x_1 + \cdots + x_n}$$

is used in the distributed ElGamal cryptosystem.

Encryption: To encrypt a plaintext message $m \in G$:

- Randomly choose an integer r from $\mathbb{Z}_q^*$;
- Computes $c_1 = g^r$;
- Computes $c_2 = g^m \cdot PK^r$.

The encrypted message is $E(m) = (c_1, c_2)$.

Decryption: A common decryption key is not computed. Each user computes and broadcasts a partially decrypted value, and the final plaintext is revealed by combining all partially decrypted values. For the ciphertext (c_1, c_2), decryption proceeds as follows:

- Each i-th user computes ${c_1}^{x_i}$;
- All users broadcast commitment of computed values $H({c_1}^{x_i})$;
- Each i-th user broadcasts ${c_1}^{x_i}$ and checks if each $c_1^{x_i}$ matches with $H(c_1^{x_i})$;
- Each user computes $\frac{c_2}{\prod_{i=1}^{n} {c_1}^{x_i}} = \frac{c_2}{c_1^{x_1 + \cdots + x_n}} = g^m$.

Finally, m can be revealed by computing a discrete logarithm.

Homomorphism

ElGamal encryption has an inherited homomorphic property [YPB14], which allows multiplication and exponentiation to be performed on a set of ciphertexts without decrypting them, such as addition homomorphic computation

$$\begin{aligned} E(m_1) \times E(m_2) &= (g^{r_1}, g^{m_1} \cdot pk^{r_1}) \times (g^{r_2}, g^{m_2} \cdot pk^{r_2}) \\ &= (g^{r_1+r_2}, g^{m_1+m_2} \cdot pk^{r_1+r_2}) \\ &= E(m_1 + m_2) \end{aligned}$$

and multiplication homomorphic computation:

$$\begin{aligned} E(m_1)^{m_2} &= (g^{r_1}, g^{m_1} \cdot pk^{r_1})^{m_2} \\ &= (g^{r_1 \cdot m_2}, g^{m_1 \cdot m_2} \cdot pk^{r_1 \cdot m_2}) \\ &= E(m_1 \cdot m_2) \end{aligned}$$

2.1.2 Proof of partial knowledge

Given a cyclic group G of a prime order q with a generator g. The secret key is x, and public key is $y = g^x$. The verifier can confirm that the ciphertext is either $E(m_1)$ or $E(m_2)$, but the verifier will never know which one is the true one [CDS94, Adi12]. The ElGamal encryption algorithm (cf. Section 2.1.1.1) is used in this protocol.

Prover

- generates a random number $r \in \mathbb{Z}_q$
- computes $E(m_1) = (c_1, c_2) = \{g^r, g^{m_1} \cdot y^r\}$
- generates random numbers $t, v_2, s_2 \in \mathbb{Z}_q$
- computes $T_0 = g^t$
- computes $T_1 = y^t$

- computes $T_2 = (g^{m_2 \cdot v_2} \cdot y^{s_2}) / {c_2}^{v_2}$
- computes $v = hash(c_1 \| c_2 \| T_0 \| T_1 \| T_2)$
- computes $v_1 = v \oplus v_2$, where $\oplus$ denotes XOR.
- computes $s_1 = r \cdot v_1 + t$
- sends $c_1, c_2, T_0, T_1, T_2, v_1, v_2, s_1, s_2$ to **Verifier**

Verifier

- verifies $v_1 \oplus v_2 = hash(c_1 \| c_2 \| T_0 \| T_1 \| T_2)$
- verifies $g^{s_1} = T_0 \cdot {c_1}^{v_1}$
- verifies $y^{s_1} = T_1 \cdot (c_2 / g^{m_1})^{v_1}$
- verifies $y^{s_2} = T_2 \cdot (c_2 / g^{m_2})^{v_2}$

If all verification tests return true, the ciphertext can be considered as encryption value of m_1 or m_2. Therefore the verifier can only confirm that the ciphertext is either $E(m_1)$ or $E(m_2)$, but cannot determine the exact value of the plaintext with certainty.

2.1.3 Proof of zero knowledge

This subsection contains prerequisites on the zero knowledge proof algorithm of [Sch91, CP92]. Suppose that plaintext message is m. In the zero knowledge proof algorithm of [Sch91, CP92] the prover computes $E(m)$ and proofs. The verifier can use the proofs only to verify that the ciphertext is encrypted from m, but cannot decrypt $E(m)$.

Further, suppose that the ElGamal cryptosystem is used, where G is a cyclic group G of a prime order q with a generator g, the secret key is x and the public key is y. Then $E(m) = (c_1, c_2) = (g^r, g^m \cdot y^r)$ (cf. Section 2.1.1.1). To verify that the ciphertext (c_1, c_2) is a correct encryption for m, the algorithm proposes verifying that c_1 and $\frac{c_2}{g^m}$ have the same exponentiation. This zero

knowledge proof is correct, because

$$c_1 = g^r \tag{2.1}$$

and

$$\frac{c_2}{g^m} = \frac{g^m \cdot y^r}{g^m} = y^r \tag{2.2}$$

Indeed, if the ciphertext (c_1, c_2) is $E(m)$, then 2.1 and 2.2 have the same exponent r. Otherwise, if (c_1, c_2) is different from $E(m)$, then it is obvious that 2.1 and 2.2 have different exponents.

Prover

- generates random number $r \in \mathbb{Z}_q$
- computes $E(m) = (c_1, c_2) = (g^r, g^m \cdot y^r)$
- generates random number $t \in \mathbb{Z}_q$
- computes $T_1 = g^t$
- computes $T_2 = y^t$
- computes $v = Hash(E(m) \| T_1 \| T_2)$
- computes $s = r \cdot v + t$
- sends c_1, c_2, T_1, T_2, v, s to **Verifier**

Verifier

- verifies if $v = Hash(E(m) \| T_1 \| T_2)$
- verifies if $g^s = {c_1}^v \cdot T_1$
- verifies if $y^s = (c_2/g^m)^v \cdot T_2$

If both verifications are passed, the verifier believes the prover's statement.

2.1.4 Distributed cryptosystem

Let G denote a finite cyclic group of prime order q in which the decision Diffie-Hellman problem is intractable. Let g be a generator in G. There are n users, all of whom agree on (G, g).

We assume there are n different users $u_1, u_2, \cdots, u_n$. Each user u_i chooses a secret value $x_i \in_R \mathbb{Z}_q$, and computes a public value g^{x_i}, where $1 \leq i \leq n$. Each u_i computes a y_i as below:

$$y_i = \begin{cases} \dfrac{1}{\prod_{k=2}^{n_v} g^{x_k}} & \text{if } i = 1 \\ \dfrac{g^{x_1}}{\prod_{k=3}^{n_v} g^{x_k}} & \text{if } i = 2 \\ \dfrac{\prod_{k=1}^{i-1} g^{x_k}}{\prod_{k=i+1}^{n_v} g^{x_k}} & \text{if } i \in [3, n_v - 2] \\ \dfrac{\prod_{k=1}^{n_v-2} g^{x_k}}{g^{x_{n_v}}} & \text{if } i = n_v - 1 \\ \prod_{k=1}^{n_v-1} g^{x_k} & \text{if } i = n_v \end{cases} \tag{2.3}$$

which is publicly computable since the computation uses all public values g^{x_i}.

Encryption: We assume the message for each u_i is m_i, and the encrypted message is

$$E(m_i, y_i, x_i) = (y_i)^{x_i} \cdot g^{m_i}$$

where $E(m_i, y_i, x_i)$ denotes the message m_i encrypted using y_i and x_i

Homomorphism addition: Anyone can compute the sum of all ciphertexts simple by multiplying all encrypted messages:

$$\prod_{i=1}^{n} E(m_i, y_i, x_i) = \prod_{i=1}^{n} (y_i)^{x_i} g^{m_i} = \prod_{i=1}^{n} (y_i)^{x_i} \times \prod_{i=1}^{n} g^{m_i} = 1 \times \prod_{i=1}^{n} g^{m_i} = g^{\sum_{i=1}^{n} m_i}$$

where $\prod_{i=1}^{n} (y_i)^{x_i} = 1$, according to [HRZ10].

2.2 Homomorphic Based Online Voting Systems

Homomorphic encryption allows a few computations (e.g. multiplication and addition) to be performed directly on the cipher texts without accessing the plain texts. Homomorphism is very useful in e-voting systems because it allows to perform tallying on encrypted votes, which protects the privacy of voters. In the voting system, if all votes are encrypted using homomorphic encryption scheme, then it makes the tallying procedure very simple [CFSY96, HS00, YO11]. Processing procedures of homomorphic tally are shown as Figure 2.1.

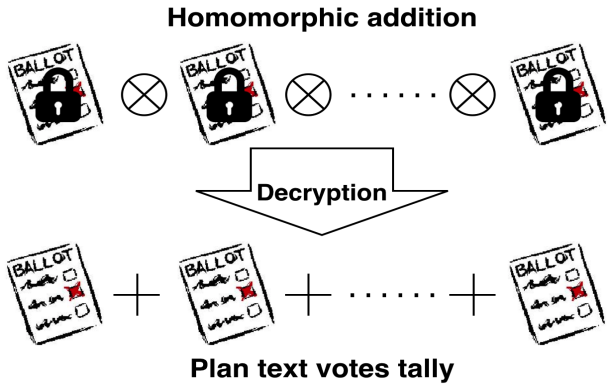

Figure 2.1: Processing procedure of homomorphism based voting scheme. All encrypted votes are multiplied, and then decrypted result is equated to the tally result of all plaintext votes.

There are several cryptosystems with inherited homomorphic addition/multiplication property, such as RSA [RSA78], Paillier [PP99] and ElGamal [ElG85]. [YO11] proposed a homomorphism e-voting system using ElGamal cryptosystem. The system has 5 stages: setup, registration, voting, tallying and verifying. In the voting stage, all votes are encrypted using ElGamal encryption algorithm before submission. As the result, all encrypted votes can be tallied by authorities without revealing the contents of each vote. At the final stage, the tallied result is decrypted and published. [HA13] proposed a secured e-voting system using Paillier encryption and blind signature to keep all votes secret during tallying. [ZPWZ14] proposed an e-voting system using RSA, which used several announcers to reveal the tallied result in order to prevent tally cheating.

Homomorphic encryption is a well-known powerful technique with many useful applications

(cf. [LVW14, AARTAD17, CLF17, LMW17, YPB14]). Recently, it has been applied to the design of online voting systems (see the next section for details). This is motivated by the need to develop advanced security systems to facilitate a broad introduction of online voting throughout the whole world. Elections conducted by paper votes are unsustainable, as they consume a lot of resources and lead to destruction of forests contributing to deterioration of climate. Recent experimental online voting in countries such as the United States, India and Brazil highlighted significant challenges that require further research to improve security guarantees in future elections.

Secure e-voting systems are required for casting votes using the Internet. Online voting systems not only increasing sustainability, but also reduce the overall cost of running elections and may increase voter participation because of the more convenient procedure, in particular, for the voters with disabilities. The study of electronic elections contributes to the more general area of privacy-preservation (cf. [ECMC16, HBB12, MNX+16, MV17, MMS+17]) and relies on secure implementations of other aspects involved in e-voting (cf. [LLS+16, NSGF16, NCS14, RFB17]). Since online voting remains vulnerable to malicious activity and hacking attacks, the design of a secure, flexible and verifiable e-voting system is a very important problem (cf. [Gre16, LK16]).

Homomorphic encryption has been used in online voting systems, for example, in [CFSY96, HS00, YO11]. The homomorphic property makes it possible to tally all encrypted ballots without decrypting them and accessing the content of any individual ballot.

Helios [Adi08] is the first web-based voting system. It used ElGamal encryption to achieve open-audit voting. Helios did do not claim any cryptographic novelty apart from that fact that, assuming that there were enough auditors, even if all the authorities fully colluded to corrupt the system, they would be unable to counterfeit the election result without a high chance of being caught. However, the security of the Helios relies on the trust of all participants in the Helios server. The security level of Helios depends on a mix-net shuffling mechanism implemented by the server. It follows that a corrupt Helios server can attempt to shuffle submitted votes incorrectly or decrypt shuffled votes incorrectly. Further, the performance results reported in

[Adi08] show that the computation time is quite long. The verification and auditing process took more than three hours on a server and a complete audit took more than four hours on voter, even though there were only 2 questions in each vote and 500 voters in total. Thus, [Adi08] provided the time achieved in only one experiment with a fixed number of voters.

All previous voting systems including [Adi08] imposed several security assumptions required on their systems. The authors of [Adi08] made an assumption that there were several honest authorities and a central honest server. If any authority is compromised, they can attempt to shuffle the votes incorrectly. Likewise, a corrupt Helios server knows the usernames and passwords of all users, and can easily authenticate and cast ballots on behalf of users.

Several improvements to Helios were made in Helios 2.0 (see [ADMP$^+$09]), which was used in a real election (UCL election). Helio 2.0 could handle 25,000 potential voters. In the UCL election, 5000 participants registered and nearly 4000 voted in each round of the election. Helio 2.0 updated the open-audit mechanism, so that it could provide more evidence of the counters' works to all voters. However, their proposed approach distributed the key-generation and decryption code among a few trusted members of the election commission. This required the trustees to be technically savvy and honest. There is no indication of the running time of any new experiments in [ADMP$^+$09].

The security assumptions in [ADMP$^+$09] were the same as in [Adi08]. In particular, a compromised server in Helios 1.0 and 2.0 could subvert all data in the ballot box near the end of the election day.

A multi-authority e-voting system introduced in [PSO11] applied the distributed ElGamal DSA with an inherited additive homomorphism property. In addition to the authorities and the voters in the election, a trusted third party was introduced and was used to distribute the shared secret key among the multiple authorities. The proposed system became receipt-free, because the encryption of each ballot now is done by the trusted third party. This means that the voters could not prove how they voted by using their encrypted vote, and the authorities of the election could

not learn the content of each vote from its encryption. However, the drawback of the system was that the third party could collude with any of the authorities and together they could recover the contents of submitted votes, which would violate the privacy of voters. The paper [PSO11] assumed that there was a trusted third party involved in the scheme to distribute the shared secret key among the authorities. There was no indication of the running time or performance analysis of any new experiments in [PSO11].

Table 2.1: Comparison of the outcomes of experiments in previous papers.

Previous papers	Presentation of experimental outcomes
[ADMP+09], [PSO11].	These papers did not include the running time of any experiments.
[Adi08], [TPLT13], [ECH12].	These papers indicated the time achieved in only one experiment with a fixed number of voters.
[Adi08], [ADMP+09], [PSO11], [ECH12], [TPLT13].	These papers did not include any line graph, bar chart or histogram with experimental outcomes.
[Gar16].	This paper presented two diagrams with line graphs comparing only the time of encryption operations.
[YYR+17].	This paper included two diagrams with line graphs comparing the performance comparison for elections with various numbers of voters.

Cobra [ECH12] was the first coercion-resistant system. It was a proof-of-concept voting system offering concurrent ballot authorization. The paper focused on the problem of coercion and vote selling. However, the proposed registration process could not be applied in practical elections, because it required each voter to register multiple times. In the performance analysis section, the authors provided concrete numbers only for a hypothetical election with 5 candidates, 10,000 registered voters, 20,000 submitted ballots and 3 trustees. This example took almost two hours of computing time on a fully parallel 8-core machine. A careful analysis of the running times of various steps of the algorithm was presented in [ECH12], but only in one experiment with a fixed number of voters. In [ECH12], an assumption was made that all authorities (trustees) should be honest. The trustees of an election authority engaged in a secure, universally verifiable protocol

implementing a ballot authorization function.

Zeus [TPLT13] was a system developed based on Helio [Adi08]. Zeus used the same workflow as Helio, but it provided more types of voting. The paper [TPLT13] assumed that the server should be always trusted. However, a potential security vulnerability of the system was that Zeus was run in a black box on a remote virtual machine. The lack of control and access to this black box virtual machine implementation created a possibility that without any control or awareness of any participants in the election process of how the black box operates, it could be subverted and confidential information could leak, or even the whole computing procedure could be incorrectly implemented. Thus, in the proposed implementation Zeus did not provide any reliable guarantee of anonymity and security to the users. Furthermore, Zeus turned out to be a computationally expensive system. It took approximately 65 minutes of computing time to handle 10000 votes by using a 16-core 2.26 GHz machine. This is the only new experiment with the running time indicated in [TPLT13].

A flexible e-voting system for online discussion forums was proposed in [Gar16], where it was assumed that there is a trusted third party (registration server). This paper presented two diagrams with line graphs comparing only the time of the encryption operations, since it considered the special case of online voting conducted as a part of continuing online debates.

In addition to all the details mentioned above, to facilitate the comparison of results of previous publications, we include Table 2.1 with a brief outline of the results of experiments in the previous papers.

2.3 Blockchain-based Online Voting Systems

A Blockchain is a public, append-only, immutable ledger maintained by a decentralised peer-to-peer network. Whilst first designed for digital currencies without trusted third parties, Blockchain technology has now moved into many fields beyond finance.

The first cryptocurrency, Bitcoin, was rated as the top performing currency in 2015 ([Des16,

Nak08]) and the best-performing commodity in 2016 ([Adi16]). It had more than 300K confirmed transactions daily in 2017 ([Blo17b]). Since the debut of Bitcoin in 2009, its underlying technique, Blockchain, has shown promising application prospects and has attracted a lot of attention from academia and industry ([Che18, LJC+17, SP18, ZMH+18]).

Blockchain technology was first introduced by the Bitcoin digital currency in 2008 [Nak08]. Bitcoin proposed a solution to securely maintain a decentralized ledger in the presence of a Byzantine failure model [LSP82], in which nodes may act maliciously.

Blockchain ledgers were originally designed to record monetary transactions, but the concept has been widened to provide support for general purpose computing. Ethereum [But15] provides a Turing complete platform for decentralized smart contracts [Blo]. Smart Contracts were first described in 1996 by Nick Szabo [Sza96] and are autonomously executing contracts written in computer code.

The Blockchain provides the following properties which make smart contracts possible:

Transparency Blockchain transactions are public and can be verified by anyone.

Immutability The transaction history of a Blockchain cannot be altered. As such, the Blockchain can be seen as an append-only database.

Trustless Participants in Blockchain transactions do not have to rely on a trusted third party for their interactions. Trust is provided by the underlying consensus protocol.

From this investigation into the protocols, the conclusion can be made that an electronic voting system must be secure, while allowing for as much transparency as possible to be a working E2E verifiable. Blockchain's [Nak08] help to achieve this level of security and transparency, while maintaining privacy and non-malleability of the transactions, which may indeed be the future of eVoting protocols. The benefits of using Blockchain is in it's decentralised nature, relatively low cost of transactions and tamper-proof properties, which play an important role if a voting system was based on this technology [Del17, Wes16].

Here we focus on Blockchain-based online voting. There are a number of existing proposals for such a system, using the Blockchain as a public bulletin board to store the voting data, such as FollowMyVote [Fol16] and TIVI [Sab16]. These proposals achieve voter privacy by involving trusted authorities that obfuscate the relation between real-world identities and keys [Fol16], or by shuffling encrypted ballots before decrypting [Sab16].

Although different, some elements from above mentioned protocols may apply to the concept of Blockchain voting. The notion of web bulletin board (WBB), where the encrypted ballots can be seen by the public members, can persist in Blockchain in the form similar to [Blo17a]. Here the blocks of transactions can be observed as well as the height of the Blockchain with any other relevant information. Although Blockchain is a promising technology, we have not found any relevant papers to date that present a protocol for online voting with Blockchains.

Examples such as FollowMyVote [Fol16] present a seemingly sound voting protocol, however without any in-depth specification to verify the security of the protocol, there is only the website information to go by. The code is open-sourced, which if compared to the notion of public cryptographic protocols over the history of private ones, indicates that it may be a secure, however, without official specification publication, or any other documentation, it is difficult to verify these claims. One other noteworthy Blockchain technology that could revolutionise electronic voting, as well as give birth to many other forms of electronic protocols is Ethereum [Woo14]. Ethereum differs from Bitcoin [Nak08] as it serves as a generic platform for creation of custom functionality in the form of contracts, while also having a slightly complex structure of transactions. The currency used by Ethereum is ether and gas, which will be discussed in more detail later in this section. However, the main difference is the fact that the contracts allow for different functionality using the Ethereum Virtual Machine (EVM), while being enforced by the peer-to-peer, decentralized way, inherent to the core structure of Blockchain. Ethereum possesses two types of accounts, which is another way of specifying types of users. Accounts are used by human entities, and potentially by other smart entities, whereas contracts are also accounts, however they are operated by

code on the EVM. Contracts are the agents that bring about the generic functionality of Ethereum mentioned above. Contacts allow one to create custom behaviour for one's Blockchain application. These applications include, and are not limited to, automatic payments or creation of custom currency, which is worthless outside of the context of the contract application. Ethereum operates on the context of states and state transitions which are brought about by the transactions. The contracts, on the other hand, have a failsafe feature which addresses the Turing Halting problem, in other words, to prevent contract code from infinite function execution, Ethereum introduces the concept of gas. Gas is consumed for each consumed resource of the contract computation. This can be each stage of contract execution or the memory used by the contract. Gas costs are dictated by the gas limit in the contract and the gas costs are deducted from the user account, who wishes to send transactions to a particular contact. This feature means that upon exceeding the gas limit for a transaction, the transaction can revert to last state and refund the user account in case of an unexpected gas limit over overflow. The contracts themselves can be used to send ether between other contracts, or to other accounts. The transaction between user accounts is simply moving ether around, which resembles the functionality of Bitcoin [Woo14, But16, Woo17].

The transactions issue receipts back to the user accounts which include information like post-transaction state, the cumulative gas used in the block containing the transaction receipt as of immediately after the transaction has happened and others. Entities such as the receipts and the world state, are stored in Merkle Patricia Trees or tries. Merkle trees are used in Bitcoin transactions and serve as a way of storing all transactions inside one hash, which can be traversed to find a particular transaction. The Merkle tree contains hashes of hashes of transactions until the root hash is obtained. In Ethereum, these tries are used as databases which stores the mappings between bytearrays of account information [Woo14, But16, Woo17].

One may wonder about the differences between Bitcoin and Ethereum, and the common question of which one is a better platform arises very often. In reality, there is no better platform out of the two, it is simply the matter of preference and functionality. Ethereum offers a set of

features that differs from that of Bitcoin. This gives a brief overview of the Ethereum protocol, and its main difference to Bitcoin. Ethereum allows the creation of powerful sub-systems, driven by custom currency and functionality with the immutable support of Blockchain. This makes it a very good candidate for different protocols, including a voting protocol.

FollowMyVote [Fol16] is generally regarded as the first organization to provide a remote voting (e-voting) solution using Blockchain technology, which exists entirely online. Firstly, voters have to establish their identification by uploading their relevant documents (e.g. driver's licence) via a downloaded application. The general idea is to use a blind signature scheme, meaning a trusted signing/verifying party would sign a blind token from a voter. Once a voter's identification is verified, he/she is able to request an online ballot and submit his/her ballot with an unblind token to the Blockchain. Furthermore, FollowMyVote allows multiple submissions from the same voter since only the most recent ballot of the voter will be counted. Moreover, FollowMyVote provides anonymous voting because the voters cannot be identified in the Blockchain. However, the voters are still clearly known to the central authority that enfranchises them. It also achieved in FollowMyVote based on elliptic curve cryptography (ECC). Each voter has 2 key pairs, one for identity verification, the other one for voting, which allows voters to verify their votes without sacrificing their right to vote anonymously.

BitCongress [Bit16] is another decentralized voting scheme that was developed based on Blockchain technology. It introduced an application called AXIOMITY as the BitCongress wallet that allows voters to participate in every aspect of the democratic process. In order to vote for any candidate, each voter should create a custom vote token as a vote, and send it to the address of that candidate. A decentralized proof-of-tally is also maintained for each voter and is updated by an election upon registering the voter's vote, which gives a profile for each voter along with his/her voting history and is used in voter verification. However, the BitCongress specification does not provide details as to exactly how this is implemented. For instance, how is this information maintained throughout the Blockchain so that it is easily accessible for

verification purposes? Moreover, anyone can register to become a voter through BitCongress allowing them to participate in democratic processes. Each address becomes associated with a Blockchain ID, allowing a person to be mapped to only one address. This ultimately requires a central authority with identity information to verify a voter and give an address in order for that voter to interact with BitCongress.

In 2017, McCorry et al. presented a smart contract implementation for the Open Vote Network that runs on Ethereum [MSH17], which is claimed to be the first implementation of a decentralized and self-tallying internet voting protocol using Blockchain. The procedure of casting a vote is similar to the procedure of generating a ring signature: each vote is generated using the voter's private key and all other voters' public keys, and the final result can only be computed by using all submitted votes. An individual vote can never be revealed unless the private key is revealed. The final result can be computed by anyone via a public accessible function. The implementation results show that the proposed protocols used with minimal setup for election, and McCorry et al. also plan to investigate the feasibility of running a larger-scale election over the Blockchain in future.

2.4 Summary

The aim of this research is to propose a decentralized publicity verifiable ranked choice online voting system, which also achieves advanced E2E verifiability. The following cryptographic techniques are required to present our system.

- ElGamal cryptosystem. ElGamal is a public key cryptosystem that satisfies the inherited additive homomorphic property. The latter allows all encrypted numbers to be added directly without decrypting them. The voters' privacy can be protected well if each cast vote is encrypted before submission, and the final outcome is still computable without decrypting any individually vote.

- Proof of Zero (partial) knowledge. Voters are able to prove the eligibility and correctness of their cast vote after encryption. This also means that any encrypted vote can be verified by the public. The proof of partial knowledge can help to develop a usable rank choice voting mechanism. By applying both the zero-knowledge proof and partial knowledge proof, we devise a new system that can reduce the dependence on the centralized authorities.

- Blockchain. A Blockchain database can be treated as a complete trusted WBB (in the centralized system), where the data can never be compromised after adding to the chain based on the decentralized feature.

Most of the existing homomorphism-based online voting systems are centralized systems. Besides, the existing previous Blockchain voting systems do not support the ranked choice voting mechanism and large-scale voting. In this research, a public verifiable online voting system is proposed and developed. It combines the conveniences achieved by applying both homomorphism and Blockchain technology, so that the system supports a flexible ranked choice voting mechanism.

Chapter 3

A PVRC Online Voting System

3.1 Motivation of PVRC

ElGamal encryption [ElG85], one of the most popular homomorphic encryptions that is used for online voting systems, inherits the addition property of homomorphism, thus allowing encrypted ballots to be tallied without decrypting individual ballots. Helio [Adi08] is the first web-based voting system to use ElGamal encryption to achieve open-audit voting. However, the verification and auditing take more than 3 hours on a server and 4 hours on a client. Several improvements to Helio were made in Helio 2.0 [ADMP+09], which was used in a real life election (University President selection of the Universite Catholique de Louvain in Louvain-la-Neuve, Belgium, 2008). Zeus [TPLT13] is developed based on Helio [Adi08], but provides more types of voting. Zeus uses the same workflow as Helio [Adi08] and is computationally expensive.

In this chapter, we propose a **Public Verifiable Ranked Choice** (**PVRC**) online voting system, which allows voters to cast and submit their electronic ballots by ranking all candidates easily according to their personal preference. Each ballot is treated as a square matrix, with each element encrypted using the ElGamal cryptosystem before submission. Furthermore, proof of partial knowledge and zero knowledge are used to verify the eligibility of ballots without accessing ballot contents. We also implement a prototype to test our proposed voting system. The

security and performance analysis indicate the feasibility of the proposed protocols.

3.2 Our PVRC Online Voting System

In this section, we present our ranked choice Internet voting system with illustrations and examples.

Voter: Each authorized voter can cast a ballot according to personal preference.

Candidate: Each candidate is a contestant in the election, and will receive points from different voters. The winner is the candidate who received the most points.

Tallying authority: Authorities in the election must take responsibility for auditing the voting process, such as verifying the identification of voters, verifying the eligibility of each submission, and revealing the winner of the election.

Public bulletin board: A secure insert-only bulletin board for publishing the information about the election, such as public keys and submitted ballots. Everyone can access the contents of the bulletin board at anytime, but no-one is able to modify or delete existing data on it.

Our system consists of the following stages: initialization stage, registration stage, ballot casting stage, ballot verification stage and tally stage. Table 3.1 provides the notations used to explain our protocols.

3.2.1 Initialization Stage

This is the beginning of an election, each authority (A_i) generates a key pair (public key pk_{a_i} and secret key sk_{a_i}). The common public key (PK) is computed using all pk_{a_i} (refer to Section 2.1), which is posted on the public bulletin board in order to encrypt each ballot before submission.

3.2.2 Registration Stage

In order to register, each voter must visit a registration station in person to present his/her valid ID (e.g. driver licence). We assume each voter owns a key pair (if not, they can generate it during

Table 3.1: Notations used in the rest of Chapter 3.

n_c:	number of Candidates
n_v:	number of Voters
n_a:	number of Authorities
V_i:	i-th voter; $i \in [1, n_v]$
pk_{v_i}:	public key of V_i; $i \in [1, n_v]$
sk_{v_i}:	secret key of V_i; $i \in [1, n_v]$
Sig_{v_i}:	digital signature of v_i; $i \in [1, n_v]$
B_i:	the ballot submitted by V_i; $i \in [1, n_v]$
$B_{j,k}^{(i)}$:	the value on position (j,k) of B_i; $j,k \in [1, n_c]$
$C_{j,k}^{(i)}$:	the encrypted value of $B_{j,k}^{(i)}$; $j,k \in [1, n_c]$
$PC_{j,k}^{(i)}$:	the proofs of $C_{j,k}^{(i)}$; $j,k \in [1, n_c]$ to prove $C_{j,k}^{(i)}$ equals either $E(0)$ or $E(1)$
$P\text{row}_j^{(i)}$:	the proofs of sum on j-th row in B_i to prove the sum equals $E(1)$
$P\text{col}_k^{(i)}$:	the proofs of sum on k-th column in B_i
A_i:	i-th Authority; $i \in [1, n_a]$
pk_{a_i}:	public key of A_i; $i \in [1, n_a]$
sk_{a_i}:	secret key of A_i; $i \in [1, n_a]$
PK:	common public key for encrypting votes
$\prod(...)$:	products of all elements
PZK{...}:	proof of zero knowledge
PPK{...}:	proof of partial knowledge

registration), which consists of a public key (pk_{v_i}) and a private key (sk_{v_i}). Once a voter's identity has been verified the voter should upload his/her pk_{v_i} to the public bulletin board.

Next, all authorized voters must sign their submissions using their sk_{v_i}, and others can verify their identities by using corresponding pk_{v_i}. In this case, Digital Signature Algorithm (DSA) is used in order to prevent adversaries submitting a ballot by impersonating any authorized voter.

3.2.3 Ballot Casting Stage

We assume there are n_c candidates, and the ballot is treated as a $n_c \times n_c$ square matrix, where different rows represent potential ranked places of different candidates. Initially, an empty ballot contains all white circles, such as (a) of Figure 3.1. Voters can rank different candidates on different rows, by turning white circles to black, such as (b) of Figure 3.1. For a valid cast ballot, each row and column contains only one black circle.

For a cast ballot, it contains only white or black circles, which are converted to "0" and

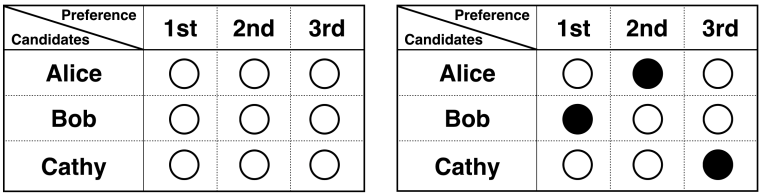

Figure 3.1: There are 3 candidates in the election. (a) is an empty ballot, (b) is a cast ballot.

"1" respectively. We use B_i to denote the ballot submitted by V_i, and $B_{j,k}^{(i)}$ denotes the value of position (j,k) of B_i, where $i \in [1,n_v]$ and $j,k \in [1,n_c]$. If the circle on $B_{j,k}^{(i)}$ is black, $B_{j,k}^{(i)} = 1$, otherwise, $B_{j,k}^{(i)} = 0$.

And then, each $B_{j,k}^{(i)}$ is encrypted using the common key PK, where we use $C_{j,k}^{(i)}$ to denote the encrypted value of $B_{j,k}^{(i)}$, such as $C_{j,k}^{(i)} = E(B_{j,k}^{(i)})$. For example, the encryption result of (b) of Figure 3.1 can be presented as follows:

$$E(B_i) = \begin{pmatrix} C_{1,1}^{(i)}, C_{1,2}^{(i)}, C_{1,3}^{(i)} \\ C_{2,1}^{(i)}, C_{2,2}^{(i)}, C_{2,3}^{(i)} \\ C_{3,1}^{(i)}, C_{3,2}^{(i)}, C_{3,3}^{(i)} \end{pmatrix} = \begin{pmatrix} E(0), E(1), E(0) \\ E(1), E(0), E(0) \\ E(0), E(0), E(1) \end{pmatrix}$$

Once $E(B_i)$ is computed, the voter must convince others the following things: each $C_{j,k}^{(i)}$ is either $E(0)$ or $E(1)$, the sums (encrypted) of each row and column equal $E(1)$. The processing details of ballot casting is shown as Algorithm 1.

3.2.4 Ballot Verification Stage

In order to prevent counting any invalid ballot into the final result, ballot verification is required as follows:

Verification of ballot's sender: The verification of each sender can be treated as the verification of the submission's signature, where the corresponding public key is published on the public bulletin board.

Algorithm 1: Ballot Casting for a voter

Input : V_i, B_i, PK, Hash function H, where $H : \{0,1\}^* \rightarrow G$
Output: encrypted ballot $E(B_i)$, all proofs of $E(B_i)$

for $j \leftarrow 1$ **to** n_c **do**
  for $k \leftarrow 1$ **to** n_c **do**
    $r, t, v_2, s_2 \in \mathbb{Z}_q$, $T_0 = g^t$, $T_1 = y^t$
    if $B_{j,k}^{(i)} = 1$ **then**
      $C_{j,k}^{(i)} = E(1) = (c_1, c_2) = \{g^r, g \cdot y^r\}$, $T_2 = y^{s_2}/{c_2}^{v_2}$
    else
      $C_{j,k}^{(i)} = E(0) = (c_1, c_2) = \{g^r, y^r\}$, $T_2 = (g^{v_2} \cdot y^{s_2})/{c_2}^{v_2}$
    end
    $v = hash(c_1 \| c_2 \| T_0 \| T_1 \| T_2)$, $v_1 = v \oplus v_2$, $s_1 = r \cdot v_1 + t$ ▷ $\oplus$: XOR
    $PC_{j,k}^{(i)} = \mathrm{PPK}\{C_{j,k}^{(i)}, T_0, T_1, T_2, v_1, v_2, s_1, s_2\}$ ▷ proof of ciphertext $C_{j,k}^{(i)}$
  end
end
for $j \leftarrow 1$ **to** n_c **do**
  $sum = C_{j,1}^{(i)} \times \cdots \times C_{j,n_c}^{(i)}$
  $t \in \mathbb{Z}_q$, $T_1 = g^t$, $T_2 = y^t$, $v = Hash(sum \| T_1 \| T_2)$, $s = r \cdot v + t$
  $P\mathrm{row}_j^{(i)} = \mathrm{PZK}_j^{(i)}\{sum, T_1, T_2, v, s\}$ ▷ proof of each row
end
for $k \leftarrow 1$ **to** n_c **do**
  $sum = C_{1,k}^{(i)} \times \cdots \times C_{n_c,k}^{(i)}$
  $t \in \mathbb{Z}_q$, $T_1 = g^t$, $T_2 = y^t$, $v = Hash(sum \| T_1 \| T_2)$, $s = r \cdot v + t$
  $P\mathrm{col}_k^{(i)} = \mathrm{PZK}_k^{(i)}\{sum, T_1, T_2, v, s\}$ ▷ proof of each column
end
digital signature: $Sig_{v_i} = Sign(E(B_i) \| PC_{j,k}^{(i)} \| P\mathrm{row}_j^{(i)} \| P\mathrm{col}_k^{(i)}, sk_{v_i})$
submit: $E(B_i)$, $PC_{j,k}^{(i)}$, $P\mathrm{row}_j^{(i)}$, $P\mathrm{col}_k^{(i)}$, Sig_{v_i}, $j,k \in [1, n_c]$

Verification of sums (each row and column): Based on the homomorphic addition, anyone is able to compute the sum (encrypted) of each row and column, and verify the value by using the corresponding proofs that are generated by the sender. The processing procedure of verification is shown as Algorithm 2.

Verification of each value (each ciphertext): Since the verification of each row and column is not sufficient to confirm a valid submission, verifying each encrypted value can be done by using the corresponding proofs that are generated during ballot casting (refer to Algorithm 1).

The processing procedure of verification is shown as Algorithm 3.

Algorithm 2: Verification of sums for each row and column in the ballot matrix

Input : $E(B_i) = C^{(i)}_{1,1}, \cdots, C^{(i)}_{n_c,n_c}, P\text{row}^{(i)}_1, \cdots, P\text{row}^{(i)}_{n_c}, P\text{col}^{(i)}_1, \cdots, P\text{col}^{(i)}_{n_c}$
Output: Valid or Invalid

for $j \leftarrow 1$ **to** n_c **do**
  $sum = C^{(i)}_{j,1} \times \cdots \times C^{(i)}_{j,n_c} = (c_1, c_2)$, $P\text{row}^{(i)}_j = \text{PZK}^{(i)}_j\{sum, T_1, T_2, v, s\}$
  if $g^s \neq {c_1}^v \cdot T_1 \;||\; y^s \neq c_2/g^v \cdot T_2$ **then**
    return Invalid ▷ Validate each row
  end
end
for $k \leftarrow 1$ **to** n_c **do**
  $sum = C^{(i)}_{1,k} \times \cdots \times C^{(i)}_{n_c,k} = (c_1, c_2)$, $P\text{col}^{(i)}_k = \text{PZK}^{(i)}_k\{sum, T_1, T_2, v, s\}$
  if $g^s \neq {c_1}^v \cdot T_1 \;||\; y^s \neq c_2/g^v \cdot T_2$ **then**
    return Invalid ▷ Validate each column
  end
end
return Valid

Algorithm 3: Verification of each ciphertext in the ballot matrix

Input : $C^{(i)}_{1,1}, \cdots, C^{(i)}_{n_c,n_c}, PC^{(i)}_{1,1}, \cdots, PC^{(i)}_{n_c,n_c}$
Output: Valid or Invalid

for $j \leftarrow 1$ **to** n_c **do**
  for $k \leftarrow 1$ **to** n_c **do**
    $C^{(i)}_{j,k} = (c_1, c_2)$
    $PC^{(i)}_{j,k} = \{C^{(i)}_{j,k}, T_0, T_1, T_2, v_1, v_2, s_1, s_2\}$
    if $\{g^{s_1} \neq T_0 \cdot {c_1}^{v_1}\} \;||$
     $\{y^{s_1} \neq T_1 \cdot (c_2/g)^{v_1}$ & $y^{s_1} \neq T_1 \cdot (c_2)^{v_2}\} \;||$
     $\{y^{s_1} \neq T_1 \cdot (c_2)^{v_1}$ & $y^{s_1} \neq T_1 \cdot (c_2/g)^{v_2}\}$ **then**
      return Invalid
    end
  end
end
return Valid

An encrypted ballot can be treated as a valid ballot only if the ballot has been verified using both Algorithms 2 and 3.

3.2.5 Tallying and Revealing Stage

Tallying is performed on each position of the ballot matrix independently. We use $C_{j,k}$ to denote the tallied result on position (j,k) of the ballot matrix, and the tallied results can be computed

according to homomorphic tallying (refer to Section 2.1), which is presented as $C_{j,k} = \prod_{i=1}^{n_v} C_{j,k}^{(i)}$.

The tallied results $(C_{1,1}, \cdots, C_{n_c,n_c})$ must be decrypted before publishing, which can only be done by a collaboration of all authorities. According to the distributed ElGamal cryptosystem, the value of (j,k) that is the final tallied ballot matrix can be presented as $D(C_{j,k}) = g^{ballots}$, therefore, the total $ballots$ on position (j,k) can be computed easily by comparing the value $g^{ballots}$ with g^1, g^2, $\cdots$, until they are equivalent.

3.3 Security Analysis on PVRC

In this section we analyse the security level of our proposed Internet voting system, under the following assumptions: 1) there are multiple authorities and at least one of them is honest; 2) the public bulletin board is secure and insert-only; 3) the Digital Signature Algorithm (DSA), the ElGamal cryptosystem, proof of zero knowledge and proof of partial knowledge are secure.

Theorem 1. *Only authorized voters are allowed to submit their ballots.*

Proof. In order to prevent adversaries from casting ballots by impersonating authenticated voters, all authorized voters must use a DSA private key to sign their submission, which can be verified by anyone using the public key of DSA. ■

Theorem 2. *Only valid ballots will be counted into the final results.*

Proof. Under our proposed system, each submission has to be verified from 2 aspects, they are 1) each ciphertext in the submission has to be either $E(0)$ or $E(1)$; and 2) the sum (encrypted) of each row and column has to be $E(1)$. In other words, a submission can be considered as valid when it has been verified by both Algorithms 2 and 3. ■

Theorem 3. *The contents of submitted ballots will never be revealed after submission.*

Proof. Under our proposed system, the content of each submission is encrypted using distributed ElGamal cryptosystem, where the encrypted key is computed using all public keys of authorities.

In this case, we assume at least one authorities is honest, meaning the content of any submission will never be revealed because the honest authority will not provide the help required to do so. ■

Theorem 4. *Any modification about any submission can be identified without difficulty.*

Proof. We assume no one can fake the DSA signature of the submission. Thus, adversaries cannot modify anything about the submission because they cannot generate a new authorized signature. Furthermore, we assume the public bulletin board is insert-only, which means no-one can submit twice or modify existing submissions. ■

Theorem 5. *The correctness of the final tallied result is voter verifiable.*

Proof. Voters are able to verify the eligibilities of all submissions, and compute the tallied result based on homomorphic addition. Furthermore, due to the tallied result being only revealed by a collaboration of all authorities, we require each authority to post computation details to the public bulletin board, allowing voters to verify the authority. ■

3.4 Performance Analysis on PVRC

This section discusses the performance of our Internet voting system. The analysis was based on the computation time of each processing step. The computation time was separated into 2 phases, client-side and server-side. All tests were performed using a 1024-bit key (p is 1024-bit), and performed on a laptop with the following specifications: CPU: 2.2 GHz Intel Core i7, Memory: 16 GB 1600 MHz DDR3.

In this case, we use t to denote the computation time of one exponentiation, where $t = 0.00012$ seconds. ElGamal encryption requires 2 exponentiations, and ElGamal decryption requires 1 exponentiation, where the division can be avoided by using an alternative method [WIK]. Thus, we use t_E and t_D to denote the computation time of encryption and decryption, respectively, where $t_E = 2t$ and $t_D = t$, approximately.

3.4.1 Performance of the voter side

The performance of client-side can be analysed by the following aspects:

Total computation time: According to the Algorithm 1, we use T_c to denote the total time spent on the client-side, where

$$T_c = t_E \times {n_c}^2 + 5t \times {n_c}^2 + 2t \times 2n_c$$

In this experiment, we tested T_c in five rounds on a laptop, according to different numbers of candidates ($n_c = 3, 5, 10, 15, 20$). The result is shown in Figure 3.2.

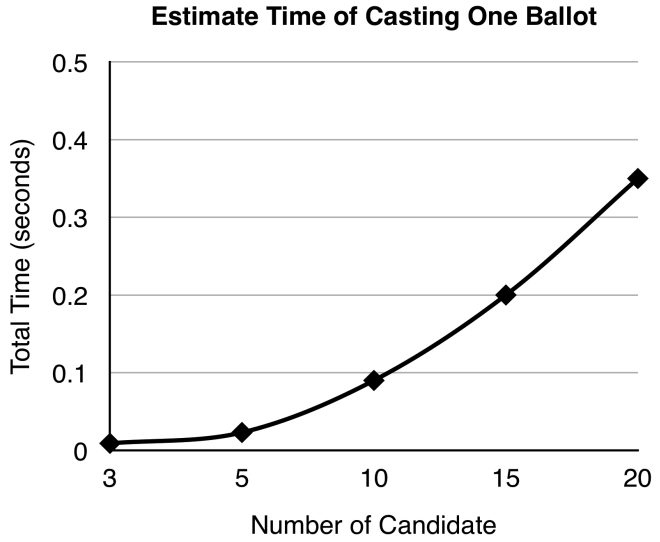

Figure 3.2: Performance of client side: Time spent encrypting a ballot when the number of candidates is 3, 5, 10, 15, 20.

From the results in Figure 3.2, we can see the time cost for encrypting one ballot is less than 0.4 seconds even if there are 20 candidates to be ranked.

Total submission size: The size of digital signature is 2048-bit, and we use S to denote the total submission size (bits) for a voter, where

$$S = 2048 \times {n_c}^2 + 7168 \times {n_c}^2 + 4096 \times 2n_c + 2048$$

and test result is shown in Figure 3.3 based on different numbers of candidates ($n_c = 3, 5, 10, 15, 20$).

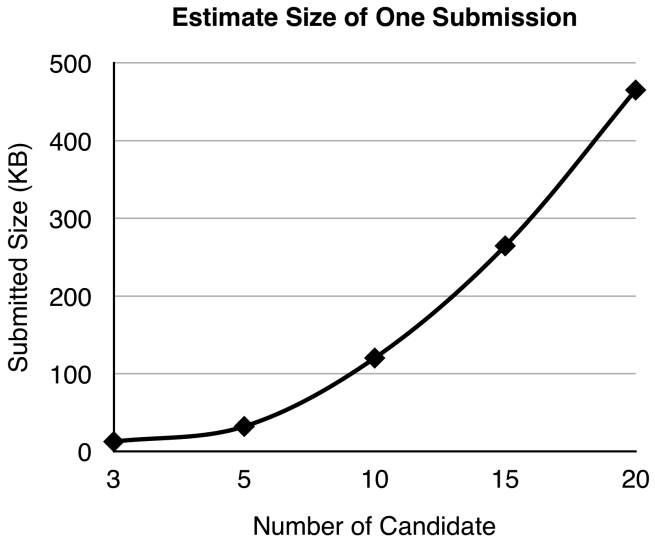

Figure 3.3: Performance of client side: The size of a cast ballot (includes all encrypted values and all proofs) when the number of candidates is 3, 5, 10, 15, 20.

From the result of Figure 3.3, we found the submission size of one ballot is less than 500KB even for a 20-candidate ballot.

3.4.2 Performance of the server side

The performance of a server can be treated as the verification of ballots. Due to the verification of each sender being equivalent to verifying the signature of each submission, this is not computationally expensive. Thus, we concentrated on the performance of Algorithm 2 and 3.

We use T_s to denote the total time spent verifying all submitted ballots, which can be presented as follows:

$$T_s = (4t \times 2n_c + 6t \times {n_c}^2) \times n_v$$

Again we tested T_s in five rounds according to different numbers of ballots (n_v = 1000, 2000, 4000, 7000, 10000). In this experiment, we assume the number of candidates is 10 ($n_c = 10$), and the result is shown in Figure 3.4.

From the results in Figure 3.4, we found the time spent verifying 10,000 ballots costs less than 15 minutes using the previously described laptop. If there are multiple super computers

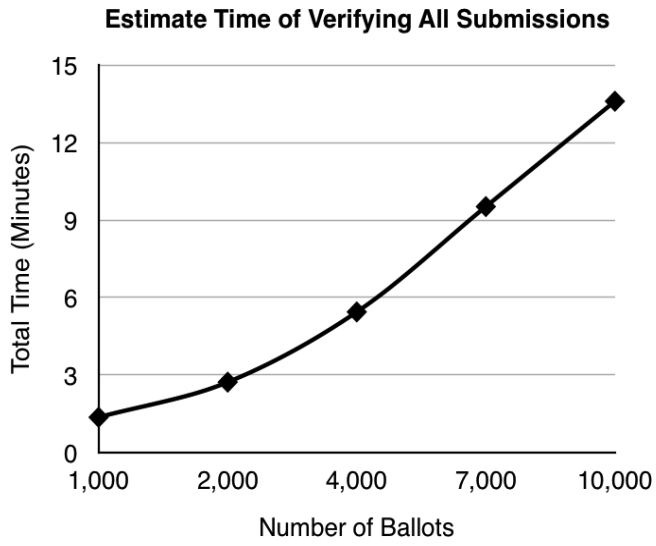

Figure 3.4: Estimate time spent of ballots' verification (by using my laptop) for 1000, 2000, 4000, 7000, 10000 ballots.

executing in parallel or a cloud computing service is available, which would be the expected case in real life, the time spent will be significantly reduced.

3.5 Conclusion

In this chapter, we have proposed a **public verifiable ranked choice** (**PVRC**) online voting system based on the distributed ElGamal cryptosystem. Under our proposed protocols, the system allows voters to rank all candidates according to their personal preferences. Although all ballots remain as ciphertexts after submission, the eligibility of each ballot has to be verified before contributing to the tallied result. Under our proposed system, voters are required to generate proofs for their cast ballots, which allows anyone to verify the eligibility of his or her submission without accessing the contents. An analytical security analysis demonstrated our proposed system is secure against external and internal adversaries. We also built a prototypical web-based voting system and tested the computational performance.

Chapter 4

A LSPVRC Online Voting System

4.1 Motivation of LSPVRC

In the previous chapter, a **PVRC** (**public verifiable online voting system**) is proposed, which allows voters to cast an electronic ballot by ranking all candidates rather than vote only for one of the candidates. However, the performance analysis indicated the feasibility of the proposed voting system with a small number of candidates. The computation cost on the voter-side will be a problem when the number of candidates is getting bigger since the ballot matrix will contain too many ciphertexts such as 10×10 when the number of candidates is 10. Thus, we aim to improve the protocol in order to solve the issue.

In this chapter, a **LSPVRC** (**Large-Scale Public Verifiable Ranked Choice**) online voting system is proposed. The computation performance on the voter side is much better than the **PVRC** system, since the ranked choice voting mechanism is re-designed, which makes this system can handle a large-scale number of voters to participate. The security and performance analyses included in this chapter not only show that our proposed protocols are feasible for practical implementations, but also demonstrate that our method has achieved significant improvements in comparison with the **PVRC** systems.

4.2 Preliminaries of LSPVRC

In the LSPVRC online voting system, the re-designed voting mechanism is inspired by the so-called approval voting also known as score voting, [BF05]. Approval voting has been used in various elections since 1987. Examples include elections conducted by some scientific and engineering societies, an econometric society and democratic state committees, [BF05]. However, to the best of our knowledge, score voting has not been applied in secure, flexible and verifiable online voting systems.

Our voting system enables voters to score all candidates and assign points to different candidates directly without any restrictions apart from the total number of available points specified by the organizers of the election. This is illustrated in Figure 4.1 where the total number of available points is equal to 6.

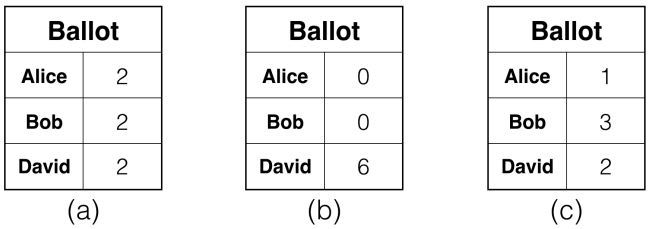

Figure 4.1: Here (a), (b) and (c) are the voting mechanism of our e-voting system when the total number of available points is equal to 6. A voter can treat all candidates equally as in (a), or support only one candidate as in (b), or rank all candidates as in (c).

Our ranked choice online voting system constructed in this chapter uses the exponential ElGamal cryptosystem due to [ElG85] (see also [AARTAD17, YPB14]). Before submission, the contents of each cast ballot are encrypted using the exponential ElGamal encryption. The additive homomorphism property of this cryptosystem, [BPS12], makes it possible to tally encrypted ballots directly without decrypting them. Our cryptosystem also includes cryptographic proofs incorporated to ensure the integrity of the voting process and to verify the validity of each ballot before it is counted.

The previous proposed online voting system (described in the preceding chapter) enables

voters to cast their ballots by ranking all candidates. The main idea behind the system proposed here is to convert each cast ballot into a square matrix, where the size of the matrix depends on the number of candidates. After that, each element in the matrix is encrypted by a verifiable homomorphic encryption algorithm. This approach, makes it possible to verify the eligibility of each submitted ballot without accessing the content of the ballot. Besides, the final result can be computed by using the additive homomorphic property without decrypting the cast ballots. The major weakness of that system is in the high cost of the computation on the voter side. It is explained by the number of exponentiations required being equal to the square of the number of candidates. It follows that the computation time can be expressed as $t_E \times {n_c}^2$, where t_E and n_c denote the computation time of a single encryption and the number of candidates, respectively. Moreover, the number of the proofs that have to be generated before submission of the ballot is equal to ${n_c}^2$, because the verification is performed based on these proofs. Creating all the required proofs significantly increases the computation time of the system proposed in the voting system (see the preceding chapter). Furthermore, it increases the size of each submission, which is made up of ${n_c}^2$ ciphertexts plus ${n_c}^2$ proofs.

The present system in this chapter is an improved and extended version of the previous system [YYR+17] as follows.

- The new data structure of a binary matrix is introduced in the present chapter. It was never considered previously. The paper [YYR+17] also used matrices, but they were different and had larger dimension.

- This innovation has significantly improved the running time of the system. The running time of our new algorithm has improved to $O(n_v \log n_v)$, as compared to $O(n_v^2)$ in [YYR+17], where n_v denotes the number of voters.

- In the present chapter, several aspects of the election system are adjusted to handle more general situation. The new system proposed in the present chapter has become applicable

in a broader class of settings and can be used for elections with less strict requirements.

All algorithms have been improved, so that this chapter contains a new "ballot generation algorithm ", "verification algorithm"and "tallying algorithm". Here are further details on improvements made here.

(1) Improved ballot generation. In [YYR$^+$17], a voter ranks all candidates to different positions and the ballot is converted to a square matrix with dimensions $n_c \times n_c$. In the present system, a voter assigns arbitrary points to different candidates, and then each point is converted to its binary representation. This representation is stored in a new binary matrix, which is not square.

(2) Improved verification algorithm. In [YYR$^+$17], the verification relied on the fact that the voters were not allowed to rank candidates to the same position in the ranked list. The system proposed in the present system allows the voters to assign the same rank to different candidates. This is why a new verification procedure has been included in this chapter.

(3) Improved tallying algorithm. In [YYR$^+$17], the tallying was based only on the additive homomorphic property. In the present system, the tallying process uses both the additive and multiplicative homomorphic properties of the cryptosystem.

(4) Improved performance. In [YYR$^+$17], the square matrix had dimensions $n_c \times n_c$. In the present system, a new binary square matrix is introduced to encode ballots. The number of rows of this matrix is n_c, but the number of columns never exceeds $2\log(n_c)$. This is why the running time of the main algorithm in the present chapter has improved to $O(n_c \log n_c)$.

4.3 Our LSPVRC Online Voting System

This section contains an overview of our ranked choice online voting system with illustrations and examples. The basic idea of our voting system is to encrypt each ballot using the common

public key of the distributed ElGamal cryptosystem. Since the exponential ElGamal satisfies the additive homomorphic property, the encrypted ballots can be directly tallied. This procedure is also known as homomorphic tallying [MMS16]. Finally, the tallied result can only be decrypted by collaboration of all authorities.

The system consists of the following stages: initialization stage, registration stage, ballot casting stage, ballot verification stage and tally stage. Table 4.1 provides the notations used to explain our protocols.

Table 4.1: Notations used in the rest of Chapter 4.

n_c:	number of candidates
n_v:	number of voters
n_a:	number of authorities
C_i:	i-th Candidate; $i \in [1, n_c]$
V_i:	i-th Voter; $i \in [1, n_v]$
A_i:	i-th Authority; $i \in [1, n_a]$
B_i:	the ballot submitted by V_i; $i \in [1, n_v]$
P:	total available points for a ballot
L_P	the number of bits of P (binary)
$B_{j,k}^{(i)}$:	the value on position (j,k) of B_i; $j \in [1, n_c]$, $k \in [1, L_P]$
$C_{j,k}^{(i)}$:	the encrypted value of $B_{j,k}^{(i)}$; $j \in [1, n_c]$, $k \in [1, L_{P_B}]$
Sig_{v_i}:	digital signature of V_i; $i \in [1, n_v]$
pk_{v_i}:	public key of v_i; $i \in [1, n_v]$
sk_{v_i}:	secret key of v_i; $i \in [1, n_v]$
pk_{a_i}:	public key of a_i; $i \in [1, n_a]$
sk_{a_i}:	secret key of a_i; $i \in [1, n_a]$
PK:	common public key for encrypting ballots
PZK{...}:	proof of zero knowledge
PPK{...}:	proof of partial knowledge

Here we list all entities that are involved in this voting system.

Voters: Each authorized voter can cast a ballot ranking all the candidates by assigning different points to different candidates according to their own preferences.

Candidates: Each candidate can be treated as a contestant in the election, and can receive different points from different submitted ballots. The candidate who received the largest total number of points is the winner of the election.

Authorities: There are multiple authorities in the election, who take responsibility for

auditing the voting process by computing the common encryption key, verifying the identification of voters for each submission, verifying the eligibility of each ballot before tallying, revealing the winner of the election.

Public bulletin board: An insert-only bulletin board, which displays all information about the election, such as public keys, all submitted ballots and final tallied result. The content of the board can be viewed by all entities. However, no one is able to modify or delete existing data on it.

4.3.1 Initialization stage

At the beginning of an election, all authorities have to generate a common encryption key (PK) that can be used by voters in order to encrypt each cast ballot before submission. Each authority (A_i) owns a key pair (public key pk_{A_i} and secret key sk_{A_i}), and (PK) is computed using the public keys (pk_{A_i}) of all tallying authorities (cf. Section 2.1.1.1). Finally, the PK is posted on the public bulletin board, which can be used by all voters.

During the common key generation, each A_i has to broadcast their pk_{A_i}. In order to prevent adversaries from replacing any public key of an authority, hash values of pk_{A_i} must be broadcast as a commitment before broadcasting the pk_{A_i}. In cryptography, the key commitment is designed such that any party cannot change the value or statement after all parties have committed to it. In our system, PK is computed using all pk_{A_i}, where a commitment means each A_i agreed to contribute their public key, which cannot be changed later. If any A_i modifies the value, the other authorities can identify this. The common key generation only commences when all the broadcast keys are authorized.

Note that the total available points for a ballot (P) must be confirmed before the election starts, the value is decided based on the number of candidates (n_c). For example, when there are 3 candidates, the value of P could be 5 or 8, as long as P is greater than n_c. Once the value of P is confirmed, all voters have to agree and use it when casting their ballots. More explanations are given in Section 4.3.

4.3.2 Registration Stage

In order to register with our e-voting system, each voter (V_i) must present their valid ID (e.g. driver licence). Once a voter's identity has been verified, he/she generates a signature key pair, which consists of a public key (pk_{v_i}) and a private key (sk_{v_i}). The pk_{v_i} is uploaded to the public bulletin board, and the sk_{v_i} is kept secret by the voter V_i.

Once a voter has completed the registration, their identity and the corresponding public key can be found on the public bulletin board. Our system requires each voter to sign their ballots using Digital Signature Algorithm (DSA), where the sk_{v_i} of voter is used to sign the voter's submission and published pk_{v_i} can be used to verify their signature Sig_{v_i}.

4.3.3 Ballot Casting Stage

This voting system allows voters to rank all candidates based on their personal preferences. Each voter can assign different numbers of points to candidates, and the winner is the candidate who receives the largest total number of points.

Voters are allowed to assign any points to any candidate. The only restriction is that the total number of assigned points must be equal to the total available points (P). To illustrate, here we look at an example where there are 3 candidates and $P = 6$. Then all options Figure 4.1(a), Figure 4.1(b) or Figure 4.1(c) are acceptable, because $2+2+2=6$ for Figure 4.1(a), $0+0+6=6$ for Figure 4.1(b), and $1+3+2=6$ for Figure 4.1(c).

After that, all assigned points are converted into binary, and the content of the cast ballot is treated as a matrix. The size of the matrix is $n_c \times L_P$, where we use n_c to denote the number of candidates, and we use L_P to denote the number of bits of P (binary). For example, the ballot illustrated in Figure 4.1(c) is converted to its binary version in Figure 4.2(b), where $P = 6$ and $L_P = 3$.

Once the ballot (B_i) is converted into its binary version, such as (b) of Figure 4.2, the content must be encrypted (using PK) before submission. In our system, each binary bit of B_i is encrypted

individually, as in Figure 4.2(c), where $B_{j,k}^{(i)}$ denotes the binary bit (0 or 1) on position (j,k) of the ballot, and $C_{j,k}^{(i)}$ denotes the encrypted value of $B_{j,k}^{(i)}$, where $j \in [1, n_c]$ and $k \in [1, L_P]$.

Once the encrypted ballot ($E(B_i) = C_{1,1}^{(i)}, \cdots, C_{n_c,L_P}^{(i)}$) is ready, there are two types of proofs to be computed and sent with the encrypted ballot. The voter (prover) must convince everyone (authorities and other voters) that each $C_{j,k}^{(i)}$ is either $E(1)$ or $E(0)$ (cf. Algorithm 4) and that the number of total number of assigned points is equal to P (cf. Algorithm 5). To this end, our system uses proof of partial knowledge (cf. Section 2.1.2) and proof of zero knowledge (cf. Section 2.1.3) to generate proofs for each $C_{j,k}^{(i)}$ and the total number of assigned points, respectively, This allows authorities and other voters to verify $E(B_i)$ without revealing the content of B_i.

Algorithm 4: Generating proofs for each element in an encrypted ballot.

Input : V_i, $E(B_i)$, PK
Output: $\text{PPKs}^{(i)}$

set PPKs = {}
for $j \leftarrow 1$ **to** n_c **do**
  for $k \leftarrow 1$ **to** L_P **do**
    PPK of $C_{j,k}^{(i)}$: prove $C_{j,k}^{(i)} = (c_1, c_2)$ is either $E(0)$ or $E(1)$ ▷ cf. Section 2.1.2
      $\text{PPK}\{(C_{j,k}^{(i)}, T_0, T_1, T_2, v_1, v_2, s_1, s_2):$
      $v_1 \oplus v_2 = H(C_{j,k}^{(i)} \| T_0 \| T_1 \| T_2),\ g^{s_1} = T_0 \cdot {c_1}^{v_1},$
      $PK^{s_1} = T_1 \cdot (c_2/g^1)^{v_1},\ PK^{s_2} = T_2 \cdot (c_2/g^0)^{v_2}\}$
    $\text{PPKs}^{(i)} = \text{PPKs}^{(i)} \cup \text{PPK}$
  end
end
return $\text{PPKs}^{(i)}$

Remark 1 The exponential ElGamal encryption is used, where $E(m) = (g^r, g^m \cdot y^r)$. For the convenience of readers, more details are given in Section 2.1.1.1.

Ballot (decimal)	
Alice	1
Bob	3
David	2

(a)

Ballot (binary)			
Alice	$B_{1,1}^{(i)} = 0$	$B_{1,2}^{(i)} = 0$	$B_{1,3}^{(i)} = 1$
Bob	$B_{2,1}^{(i)} = 0$	$B_{2,2}^{(i)} = 1$	$B_{2,3}^{(i)} = 1$
David	$B_{3,1}^{(i)} = 0$	$B_{3,2}^{(i)} = 1$	$B_{3,3}^{(i)} = 0$

(b)

$C_{1,1}^{(i)} = E(0)$	$C_{1,2}^{(i)} = E(0)$	$C_{1,3}^{(i)} = E(1)$
$C_{2,1}^{(i)} = E(0)$	$C_{2,2}^{(i)} = E(1)$	$C_{2,3}^{(i)} = E(1)$
$C_{3,1}^{(i)} = E(0)$	$C_{3,2}^{(i)} = E(1)$	$C_{3,3}^{(i)} = E(0)$

(c)

Figure 4.2: (a) is a ballot B_i cast by a voter V_i, (b) is a binary version of B_i, (c) is the encrypted version $E(B_i)$.

Algorithm 5: Generating proofs for the total number of assigned points of an encrypted ballot.

Input : $V_i, E(B_i), PK, P$
Output: $\text{PZK}^{(i)}$

set PZK = {}
set $P_{B_i} = E(0)$
for $j \leftarrow 1$ **to** n_c **do**
    set $t = 1$
    for $k \leftarrow L_P$ **to** 1 **do**
        $P_{B_i} = P_{B_i} \times (C^{(i)}_{j,k})^t$
        $t = t \times 2$
    end
end
generates proof of P_{B_i}: prove $P_{B_i} = (c_1, c_2)$ is $E(P)$ ▷ cf. Section 2.1.3
$\text{PZK}^{(i)}\{(P_{B_i}, T_1, T_2, s) : v = hash(P_{B_i} \| T_1 \| T_2), g^s = {c_1}^v \cdot T_1, PK^s = (c_2/g^P)^v \cdot T_2\}$.
return $\text{PZK}^{(i)}$

Remark 2 The proof of partial knowledge and the proof of zero knowledge in Algorithms 4 and 5 are denoted by PPK $\{\alpha, b, \cdots : \alpha, \beta, \ldots\}$ and PZK$\{\alpha, b, \cdots : \alpha, \beta, \ldots\}$, respectively. Here $\alpha, b, \ldots$ are the proofs generated by the prover, and $\alpha, \beta, \ldots$ are the conditions satisfied by $\alpha, b, \ldots$. The algorithm PPK verifies that a ciphertext $E(m)$ is the encryption of one of the multiple values $m_1, m_2, \ldots$ without decrypting it. Likewise, PZK verifies (without decryption) that the ciphertext $E(m)$ is the encryption of m. Both PPK and PZK never reveal the content of the ciphertext. The algorithms PPK and PZK are well-known. For the readers' convenience, more details on PPK and PZK are included in Sections 2.1.2 and 2.1.3, respectively.

When the ballot has been cast, a digital signature (Sig_{v_i}) of the voter (V_i) is generated and sent with $E(B_i)$ and all proofs to the server. In this case, DSA is used, which means that Sig_{v_i} is generated by using sk_{v_i} and can be verified by using pk_{v_i}, as explained in the next subsection.

To summarize, each submission consists of the following: encrypted contents of a cast ballot ($E(B_i)$), proofs of each ciphertext ($\text{PPKs}^{(i)}$), proofs of total number of assigned points for the ballot ($\text{PZK}^{(i)}$) and a digital signature (Sig_{v_i}).

4.3.4 Ballot Verification Stage

The contents of each submission are posted on the public bulletin board, including all encrypted values, all proofs and the digital signature. However, to prevent tallying any invalid ballot to the final result, the verification of each submission is a necessary and crucial step. It consists of the following three verifications: (1) verifying whether the sender of the submission is authorized, (2) verifying whether each encrypted element of the cast ballot ($E(B_i)$) is either $E(1)$ or $E(0)$, and (3) verifying whether the total number of assigned points of the $E(B_i)$ is equal to P.

In our system, each ballot is tallied to the final result only if its submission has been validated in the following three verification steps.

(1) Verify the sender of each submission: In order to prevent unauthorized people impersonating authorized voters, we require each voter to sign their submission by using their private key (sk_{v_i}) based on the DSA algorithm. This means that the signature can be verified by using the voter's public key (pk_{v_i}).

Since the pk_{v_i} of every authorized voter is posted on the public bulletin board once he/she is successfully registered, it follows that anyone can verify whether a subsequent submission is sent by an authorized voter or not. To this end it suffices to verify its signature. For example, if $VerifySignature(Sig_{v_i}, pk_{v_i})$ is true, then the identification of the sender holds true and the sender is an authorized user; otherwise, the submission is discarded.

(2) Verify each encrypted element of the cast ballot: Each ballot is treated as a binary matrix during ballot casting, as in Figure 4.2(b). It means that each element of a valid submitted ballot must be either $E(1)$ or $E(0)$, as in Figure4.2(c).

Our system generates proofs for each voter submission based on the well-known proof of partial knowledge protocol (cf. Section 2.1.2). This proof can be used to verify each encrypted element of the ballot without decryption. The verification of each element consists of two statements, both of which have to be confirmed as true. If either of these two statements is not true, then the whole ballot $E(B_i)$ cannot be counted in the final result. Only if all elements of the $E(B_i)$ are confirmed

as valid, then $E(B_i)$ is considered as valid.

(3) Verify the total number of assigned points of the cast ballot: The total available points (P) of a ballot has been confirmed before the election. It is necessary to verify that the total number of assigned points for each ballot are equal to P in order to prevent a voter assigning more points to their ballots. In our system, we require each voter to generate proof for the total number of assigned points for their ballots based on zero knowledge proof procedure explained in Section 2.1.3.

In this case, anyone is able to compute the total assigned points of any submitted ballots according to lines 1 to 8 of Algorithm 5. Then the self-computed value can be verified by using the voter-generated proof without decrypting anything. For convenience of the readers, more details on zero knowledge proof procedure are given in Section 2.1.3.

4.3.5 Tallying and Revealing Stage

In our system, each cast ballot is encrypted by using ElGamal encryption algorithm, which allows all encrypted ballots to be tallied directly without decrypting the contents of any ballot (cf. Section 2.1.1.1).

The tallying is performed on each row of the binary ballot matrix, because each row denotes the received points of a particular candidate. In this case, we use P^{c_j} to denote the final tallied result for Candidate C_j, where $i \in [1, n_c]$. In order to tally P^{c_j} for C_j, the assigned points of each encrypted ballots are converted to decimal values by doing the exponentiation computation (cf. Section 2.1.1.1). Once all encrypted ballots are converted as encrypted decimal ballots, the final result can be tallied based on different rows of all ballots. An illustration of tallying two encrypted ballots $E(B_1)$ and $E(B_2)$ is shown in Figure 4.3 where there are 3 candidates C_1, C_2 and C_3.

The procedure of tallying all valid encrypted ballots is described in Algorithm 6.

Since the final tallied results ($P^{c_1}, \cdots, P^{c_{n_c}}$) for all candidates remain stored as ciphertexts, all of them have to be decrypted before publication. According to the distributed ElGamal

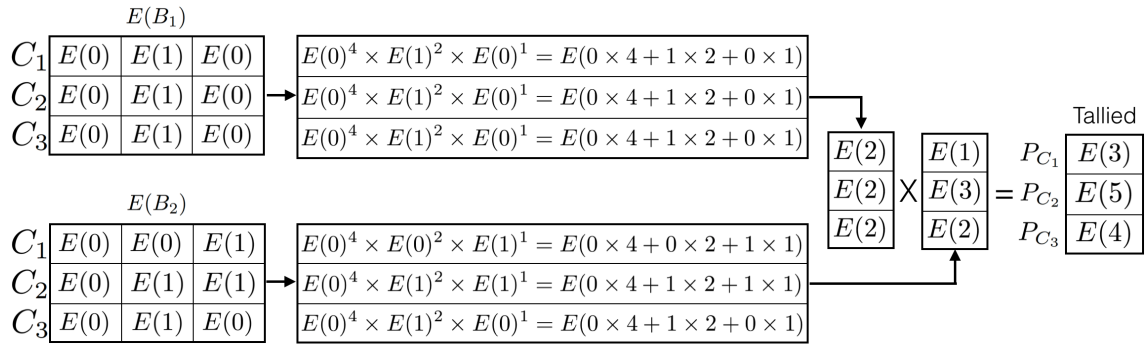

Figure 4.3: An illustration of tallying two encrypted ballots.

Algorithm 6: Tally all valid encrypted ballots.

Input : all encrypted ballots $E(B_1), \cdots, E(B_{n_v})$
Output: $P^{c_1}, \cdots, P^{c_{n_c}}$

for $j \leftarrow 1$ **to** n_c **do**
  set $P^{c_j} = E(0)$
end
for $i \leftarrow 1$ **to** n_v **do**
  for $j \leftarrow 1$ **to** n_c **do**
    set $t = 1$
    for $k \leftarrow L_P$ **to** 1 **do**
      $P^{c_j} = P^{c_j} \times (C_{j,k}^{(i)})^t$ ▷ cf. Section 2.1.1.1
      $t = t \times 2$
    end
  end
end
return $P^{c_1}, \cdots, P^{c_{n_c}}$

cryptosystem (cf. Section 2.1.1.1), the decryption procedure can only be done by collaboration of all authorities $(A_1, \cdots, A_{n_a})$, which requires each A_i to compute a partially decrypted value (cf. Section 2.1.1.1), such as $c_1^{sk_{A_i}}$ and broadcast it to the others. In order to prevent a compromised A_i from any bad actions, such as somehow computing the value incorrectly, our system requires each A_i has to generate proof of zero knowledge (cf. Section 2.1.3) in order to prove that the broadcast value is computed correctly, such as the exponent of the broadcast value (e.g. $c_1^{sk_{A_i}}$) must be the same as the exponent of its public key (e.g. $pk_{A_i} = g^{sk_{A_i}}$).

After decryption, the tallied result of each candidate must be revealed (cf. Section 2.1.1.1) and published on the bulletin board.

4.4 Security analysis on LSPVRC

This section is devoted to a theoretical security analysis of our system. Note that none of the previous related papers provided a formal security model. They only included a description and an informal security discussion of their systems: see Section 1. Our system relies on the ElGamal cryptosystem and several basic cryptographic protocols, which are presented in Section 2 and have reliable published proofs of their security. This is why here we include brief self-contained proofs of several theorems, which demonstrate that our proposed system fulfils all the security requirements.

Theorem 6. *If the digital signature algorithm (DSA) is unforgettable, no one is able to submit a ballot by impersonating another voter.*

Proof. In order to prevent adversaries from casting ballots by impersonating authenticated voters, we use a digital signature algorithm (DSA), which requires each voter V_i to have a key pair (public key pk_{v_i} and private key sk_{v_i}). The key pair is generated only if a voter is successfully verified during the registration. Then the pk_{v_i} of each verified voter is posted on the public bulletin board, and the voter is responsible for keeping their private key secret.

Once the election starts, each authorized voter signs their cast ballot by using their sk_{v_i}, and submits the encrypted ballot $E(B_{v_i},PK)$ (which indicates the cast ballot B_{v_i} by voter V_i is encrypted using the common public key PK) along with their signature Sig_{v_i} to the public bulletin board, such as

$$\{E(B_{V_1},PK), \text{corresponding proofs}, Sig_{v_1}\}$$

on the bulletin board. Others are able to verify the eligibility of each submission by verifying the Sig_{v_i} using the corresponding pk_{v_i} of V_i, where all pk_{v_i} of the successfully registered voters should be published on the public bulletin board.

In our protocols, no sk_{v_i} of voters is ever transferred. This means that only the voters themselves know their private keys. Therefore, no one is able to fake a voter's signature without

the private key, and an adversary cannot submit a ballot by impersonating an authorized voter. ■

Theorem 7. *Only one submission from each voter can be stored on the server.*

Proof. In our voting system, only the content of a submitted ballot is encrypted, the identification of the voter is in plaintext and can be viewed by everyone. For instance, in the extended previous example

$$V_1 \rightarrow \{E(B_{V_1}, PK), \text{corresponding proofs}, Sig_{v_1}\}$$

where the V_i is not encrypted.

Everyone is able to see the voter has submitted their ballots (e.g. voter's name or ID V_i is plaintext), but no one is able to discover how he/she voted.

Thus, multiple-voting detection is achieved by our system, because it is clear that it can always detect whether a voter has previously submitted a ballot: the voter id V_i for each submission is done on plaintext in the public bulletin board.

Furthermore, according to the requirements in the real-life case, our system can keep the first submission of each voter or replace the previous submission for each voter before the deadline. ■

Theorem 8. *If the ElGamal cryptosystem is semantically secure and at least one of authorities is honest, then the contents of ballots will not be revealed during ballot submission.*

Proof. Every cast ballot is encrypted before submission. We use the distributed ElGamal cryptosystem (cf. Section 2.1.1.1), which inherits the homomorphic property from the standard ElGamal system and is semantically secure as long as at least one of authorities is honest.

In our system, no one can reveal the contents of the ballots because of the following three reasons. First, all the submitted ballots remain in encrypted form as ciphertexts $E(B_{v_i}, PK)$ all the time. The homomorphic property makes it possible to add all $E(B_{v_i}, PK)$ without decrypting them. Second, there is no relationship between the ciphertexts $E(B_{v_i}, PK)$ and the corresponding

plaintexts B_{v_i}, since the cryptosystem employed is probabilistic. It applies random numbers so that the ciphertext $E(B_{v_i},PK)$ can take on different values even when the encryption $E(B_{v_i},PK)$ is computed with the same B_{v_i} and PK. Third, the decryption must be done via collaboration of all authorities A_i as in the expression

$$D(E(B_{v_i},PK),sk_{A_i},\cdots,sk_{A_{na}}),$$

where B_{v_i} cannot be correctly computed if any of the sk_{A_i} is missing. As mentioned above, we assume that at least one of the authorities is honest and will not help others to do the decryption. Thus, the contents of each ballot remain secure. ■

Theorem 9. *If the partial knowledge proof protocol (cf. Section 2.1.2) and the zero knowledge protocol (cf. Section 2.1.3) o not reveal knowledge, contents of ballots will not be revealed during ballot verification.*

Proof. In order to prevent counting invalid ballots to the final result, each encrypted ballot has to be verified before tallying. In our system, each encrypted ballot will be verified from 2 aspect: verify each encrypted element of the ballot, and verify the total number of assigned points of the ballot. Neither of the verification algorithms will reveal the voter's privacy, the analysis is shown as follows:

Verifying each element of a ballot: For an encrypted ballot, each element in the ballot is encrypted individually. We also require the voter to generate proofs of each element, which is computed based on proofs of partial knowledge (cf. Section 2.1.2).

The proof of each encrypted element $PPK\{...\}$ is generated based on the proof of partial knowledge, which consists of T_0,T_1,T_2,v_1,v_2,s_1 and s_2, where v_2 and s_2 are random values, and T_0,T_1,T_2,v_1,s_1 are computed by using random values t,v_2,s_2. Furthermore, the proof of partial knowledge is given in [Adi08], which will not reveal the content of the original plaintext.

By using $PPK\{\cdots\}$, everyone can verify any element without decryption. The verifier(s) could only know if an element is either $E(0)$ or $E(1)$, but cannot determine the exactly value of the element is 0 or 1, and so the content of the ballot did not be revealed.

Only if all elements of a ballot are verified as valid, can the ballot be regarded as valid. In this case, we assume/assert the proof of zero knowledge is secure and will not reveal the information of the content. Thus, our protocol is secure and will not revealing voter privacy.

Verifying the total number of assigned points of a ballot: The total number of assigned points of a ballot can be computed according to lines 1 through to 8 of Algorithm 5. We also require each voter to generate the proof for the total assigned points in order to convince the verifier(s) that the total number of assigned points is equal to the total available points.

According to the proof of knowledge $PZK\{...\}$ (cf. Section 2.1.3), the self-computed total assigned points by verifier(s) can be verified by using voter-generated proofs without decrypting the self computed value.

The $PZK\{...\}$ consists of T_1, T_2, v and s, where all of these elements are generated by using another random number t, which is unrelated to $PZK\{...\}$ and $PPK\{...\}$. This means that nobody can derive any useful information from $PZK\{...\}$. The usability of $PZK\{...\}$ is proved in [YYR+17].

Moreover, even if the total number of assigned points of the ballot is decrypted, the voting preferences of the voter will not be disclosed and the voters' privacy will not be revealed because the total number of assigned points of each ballot should equal the total available points (public information), otherwise the ballot will be considered invalid and be discarded. ■

None of the steps "verify each element of the ballot"and "verify the total assigned points of the ballot"require decryption. It follows that no one is able to reveal the content from the ciphertext without decryption because of the assumption that the ElGamal cryptosystem is secure. Furthermore, the decryption needs the collaboration of all authorities, and we assume that an honest authority will never commit bad actions. This means that decryption will never be executed in cases where it is not required. To sum up, the contents of the ballot and the privacy

of voters remain secure in our verification protocols.

Theorem 10. *Integrity of ballots is secured after submission.*

Proof. We require voters to sign their ballots by using their private keys based on DSA, and we assume that all voters do not share their private keys with anyone else, to ensure that nobody can fake anyone else's signature.

The integrity of ballots can be protected in the following aspects: 1) Voters are able to save all contents (the ciphertexts) of the submissions as original receipts. Once the signature is verified, all contents are posted to the public bulletin board: voters can easily detect if their submissions are modified by comparing the original receipts and the published contents. 2) Once all contents of the submissions are posted on the public bulletin board, everyone can verify the integrity of any ballot by verifying the signature of the submission, because the signature is computed based on the content of the ballot (cf. Section 4.3). ■

Theorem 11. *Invalid ballots will be detected and discarded before tallying.*

Proof. To prevent any invalid ballot from being tallied in the final result, each ballot has to undergo two verifications to check the total number of assigned points of a ballot and each element in the ballot.

Verifying each element in a cast ballot: In our protocol, the verifiers are able to verify that any element of a submitted ballot is either $E(0)$ or $E(1)$, which is used and proved in [Adi08, ADMP$^+$09]. Once the ballot is verified as consisting only of 0s or 1s in plaintext, it follows that the voter did not assign negative points to any candidate because the value (decimal) of a binary that contains only occurrences of the digits 1 and/or 0 must be greater than or equal to 0. Once all elements are verified as valid, all candidates of the ballot receive non-negative points.

Verifying the total number of assigned points in a cast ballot: An honest voter can assign any point to any candidate according to their personal preferences, but the sum of all

assigned points P_{B_i} should be equal to the total available points P for a ballot, which is confirmed before the election begins.

In our system, the total number of assigned points of a ballot can be computed according to lines 1 to 8 of Algorithm 5, and verified by using voter-generated proof (cf. Section 4.3.4). Once the total assigned points is verified as valid, it follows that the voter did not assign excess points to the ballot, which is used and proved in [YYR+17]. ■

To sum up, a ballot can be considered as valid and contribute to the final result if and only if 1) the voter did not assign negative points to any candidate and 2) the voter did not assign excess points for the ballot. Otherwise, the ballot will be discarded.

Theorem 12. *Voters are able to verify the integrity and eligibility of their ballots and the correctness of final tallied result.*

Proof. In our system, all contents (encrypted ballot, all proofs and signature) of each submission are be posted on the public bulletin board, where they can be accessed by anyone. We assume that the bulletin board is secure, and it is "append-only". Voters can use the contents published on the public bulletin board to verify the following. First, they can verify the integrity of their own submission, because in the ElGamal cryptosystem each ciphertext $(g^r, g^m \cdot y^r)$ can be verified, since g and y are public values, m is the voting message and each r is known to the corresponding voter. Thus, each voter can verify that the ciphertext is a correct encryption of the ballot. Second, all participants can verify the eligibility of submissions of other voters, because the "verification of each submission uses only encrypted submissions available to all users as part of our system (see Section 4.3.4).

Furthermore, the final tallied result (ciphertext) can also be computed by anyone, because all encrypted ballots have to be tallied based on additive and multiplicative homomorphic property of ElGamal cryptosystem (cf. Section 2.1.1.1), which is also a publicly accessible algorithm. The final decryption is performed by the collaboration of all authorities, where each authority computes

the partial decryption value by using their secret keys. In our system, we require that each authority A_i be able to convince other users that the computed value $(g^r)^{sk_{A_i}}$ is the partial decryption of ciphertext $(g^r, g^m \cdot y^r)$ using proof of zero knowledge, where the value has the same exponentiation as $g^{sk_{A_i}}$, the public key of the authority. ■

4.5 Performance Analysis on LSPVRC

The analysis of performance of each processing step is presented in a separate subsection below. All tests were performed using a 1024-bit key (p and q are 1024-bit). All tests were performed on a laptop with the following specifications: 2.8GHz quad-core Intel Core i7 (Turbo Boost up to 4.0GHz) with 6MB shared L3 cache and with 16GB of 1600MHz DDR3L onboard memory. We used a high performing implementation from libgmp via the gmpy2 python module [GMP]. We use t to denote The computation time of one exponentiation is denoted by t. For the computer used in our experiments, we have $t = 0.00012$ seconds.

ElGamal encryption requires two exponentiations, and ElGamal decryption requires one exponentiation (cf. Section 2.1.1.1), where the division can be avoided by using an alternative method [ElG85]. In this case, we use t_E and t_D to denote the computation time of encryption and decryption, respectively, where $t_E = 2t$ and $t_D = t$, approximately.

4.5.1 Performance of the voter side

On the client-side, each voter should cast a ballot and submit it to the server. In order to prevent the voting preferences of each ballot being revealed after submission, we require each voter to encrypt their cast ballots, where every element in the ballot must be encrypted.

In this case, we set $P = 2n_c$, so that the total available points P is equal to twice the number of candidates n_c. For example, if there are 10 candidates ($n_c = 10$) in the election, each voter has 20 available points ($P = 2n_c = 20$) for their cast ballot. Since all elements in the cast ballot have to be encrypted, the larger the number of candidates participating, the more encryption processing

time is required and the larger the submission grows. Therefore, the performance of the voter side can be summarized in two aspects: total computation time and total submission size.

Total computation time. We use the well-known Digital Signature Algorithm (DSA) to sign each ballot before submission. The processing time of signing is approximately equal to the time of one exponentiation, that is t. Thus, according to Section 4.3, the total computation time for a voter can be presented as the total computation time of $E(B_i)$, PPKs$^{(i)}$, PZK$^{(i)}$ and Sig_{v_i}, which can be expressed as

$$2t \times n_c \times L_P + 5t \times n_c \times L_P + 2t + t.$$

In this experiment, we test the total time spent for encrypting one ballot in five rounds on the laptop, according to different numbers of candidates (n_c =3, 5, 10, 15, 20) and different total available points ($P = 2n_c$ = 6, 10, 20, 30, 40). Hence the numbers of bits (L_P) for each available points are L_P = 3, 4, 5, 6, 6, respectively. The result is shown in Figure 4.4.

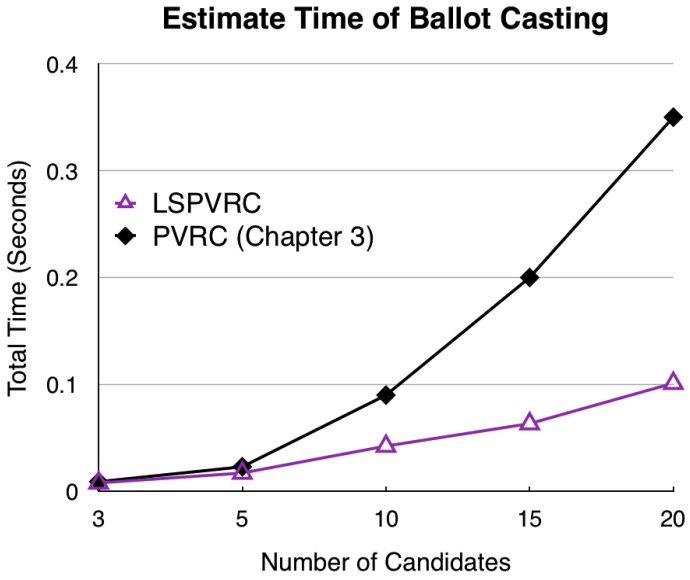

Figure 4.4: Estimate total time spent casting a ballot when the number of candidates (n_c) is 3, 5, 10, 15, 20, and the total available points (P) is 6, 10, 20, 30, 40. The performance of the system considered in [YYR$^+$17] is also shown.

The computation time presented in Figure 4.4 does not include the thinking time of the voters. It represents only the computation time of the algorithm converting all plaintext ballots cast by

the voters to ciphertext ballots and tallying them.

From the results in Figure 4.4, we can see that the time cost for casting a ballot is approximately 0.1 seconds even if there are 20 candidates in the election.

Total submission size. In our system, the size of a ciphertext that is encrypted by ElGamal is 2048-bit (cf. Section 2.1.1.1), and the size of proofs for each ciphertext is 7168-bit (cf. Section 2.1.2). Furthermore, the size of proof of total number of assigned points is 5120-bit (cf. Section 2.1.3) and the size of digital signature is 2048-bit. Thus, according to Section 4.3, the total submission size for a voter can be summarized as the total size of $E(B_i)$, $\text{PPKs}^{(i)}$, $\text{PZK}^{(i)}$ and Sig_{v_i}:

$$2048 \times n_c \times L_P + 7168 \times n_c \times L_P + 5120 + 2048.$$

In this experiment, we test the total submission size (including all encrypted elements, all proofs and a digital signature) for one cast ballot in five rounds on the laptop, according to different numbers of candidates ($n_c = 3, 5, 10, 15, 20$) and different total available points ($L_P = 3, 4, 5, 6, 6$). The result is shown in Figure 4.5.

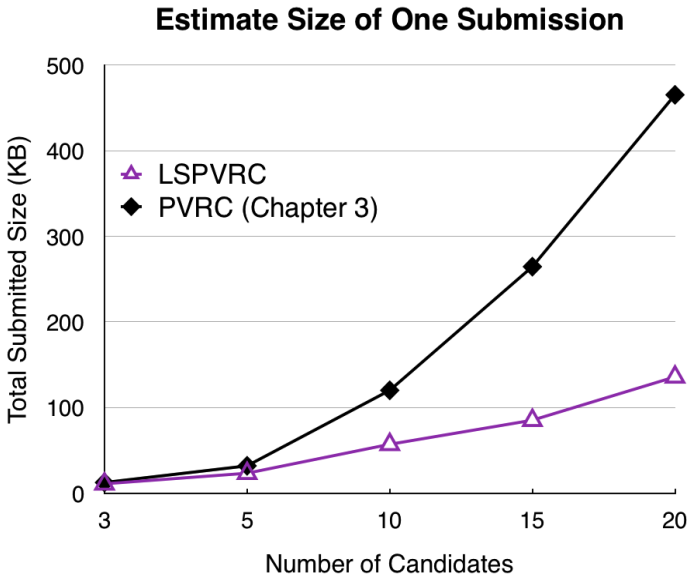

Figure 4.5: Estimate total size of one submission (including all encrypted values and all proofs) when n_c is 3, 5, 10, 15, 20, and P is 6, 10, 20, 30, 40. The performance of the system considered in [YYR+17] is also shown.

From these results, we found that the size of one submission is less than 150KB even for a 20-candidate ballot including all encrypted values and all proofs. It may be hard for the voters to compare too many candidates, and so much larger numbers of candidates seldom occur in elections. According to the Speedtest Global Index [36], in December 2017 the global average internet speed was equal to 21.25 Mbps download with 8.88 Mbps upload for mobile internet connections, and 40.71 Mbps download with 20.22 Mbps upload for fixed broadband connections, respectively. The Speedtest Global Index [Spe17] evaluates and ranks the mobile and fixed internet connection speeds from around the world on a monthly basis. This demonstrates that the submission size of a cast ballot in our system is small enough to be submitted over the internet without delays for the voters. Note that the internet connection speed is rapidly increasing in the whole world.

Besides, in implementing practical applications the running time of our system can be improved further fine-tuning the system, for example, if the secure method for handling the encryption proposed in [KKJ$^+$16] is applied. Other efficient applications of encryption have been developed, for example, in [BSE17, BHRC17, PRB$^+$17, XMY$^+$17].

4.5.2 Performance of the server side

The performance of the server can be summarized from two aspects: verification of senders and verification of ballots. The verification of each sender is equivalent to verifying the digital signature of each submission, which does not cost much computation time. Therefore, we concentrate on the performance of ballot verification, which has two parts: the time of verifying each element of each ballot and time of verifying the total assigned points of each ballot. The total computation time on the server side is equal to

$$6t \times n_c \times L_P \times n_v + 4t \times n_v$$

Our experiments determined the total time spent on verifying all submitted ballots in five rounds according to different numbers of voters ($n_v = 1000, 2000, 4000, 7000, 10000$). In this experiment,

we assume that the number of candidates is 10 ($n_c = 10$), and the total available points for a ballot is 20 ($P = 20$). Thus, the result is shown in Figure 4.6.

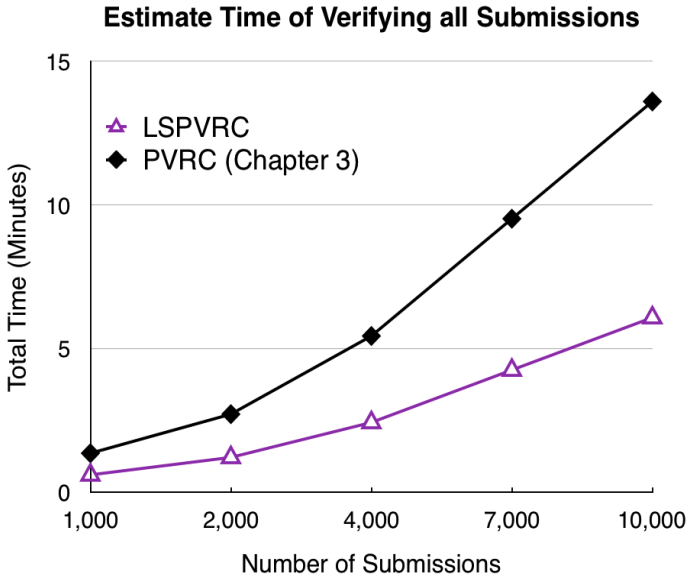

Figure 4.6: Estimate of the total time required for the verification of all ballots for 1000, 2000, 4000, 7000, 10000 voters, in the case of 10 candidates in the election. The performance of the system considered in [YYR+17] is also shown.

From the results in Figure 4.6, we found that the time spent of verifying 10,000 ballots took approximately 6 minutes using the same laptop (the specifications of which are provided in the beginning of this section). Furthermore, in our protocols, all verifications can be done by an individual without collaboration with others. Thus, all submitted ballots can be divided into multiple groups for simultaneous verification by different authorities. This can significantly reduce the time spent. For example, if there are 10 authorities in the election, then in practice the total verification time is approximately 10 times faster than the corresponding result presented in our diagrams, because in practice all authorities can work in parallel at the same time. Moreover, in the case of actual elections, it is natural to expect that cloud computing services will be available or multicore super computers can be used to execute the algorithms in parallel. This means that the running time will be further reduced.

The results presented in Figures 4.4, 4.5, and 4.6 clearly demonstrate that our system has

achieved a substantial improvement in the running time in comparison to [YYR+17]. It is easy to see this, because all the experimental results of [YYR+17] are also included as line graphs in Figures 4.4, 4.5, and 4.6.

4.6 Conclusion

In this chapter, we propose a **LSPVRC** online voting system, which is an extended version of the **PVRC** system (in **Chapter 3**). The **LSPVRC** online voting system allows the voters to cast their ballots by assigning arbitrary numbers of points to different candidates. This means that the voters can assign equal points to different candidates, and they are also allowed to assign different points to different candidates. Our system incorporates the distributed ElGamal cryptosystem. Each cast ballot is encrypted before submission and remains encrypted at all times. The additive homomorphic property of the exponential ElGamal cryptosystem enables effective processing of the ciphertexts during these procedures. Furthermore, the eligibility of voters and their submissions can be verified by anyone without the contents of the ballots being revealed. The security and performance analysis not only confirm the feasibility of our online voting system for practical elections but also demonstrate that it has achieved significant improvements over other systems considered previously.

Chapter 5

A SDPVRC Online Voting System

5.1 Motivation of SDPVRC

In the previous Chapter, a **LSPVRC** (**Large-Scale Public Verifiable Ranked Choice**) online voting system is proposed by using a score based voting mechanism. However, there is a limitation of the system, which is that we have to assume that there are multiple tallying authorities in the system and at least one of them is honest since otherwise, the system is not secure if all authorities colluded. In this Chapter, we aim to solve this issue.

In this chapter, a **SDPVRC** (**Semi-Decentralized Public Verifiable Ranked Choice**) online voting system is proposed. We argue that Blockchain technology, combined with modern cryptography can provide the transparency, integrity and confidentiality required from reliable online voting. Furthermore, we present a decentralized online voting system implemented as a smart contract on the Ethereum Blockchain. The system only needs one tallying authority, and he/she can never reveal any voter's privacy.

5.2 Preliminaries of SDPVRC

We aim to propose a **SDPVRC** online voting system using a smart contract deployed on Ethereum. The system reduces the responsibilities of election authorities to a minimum and allows candidate ranking, instead of just voting for one candidate [MSH17]. The system's voting mechanism is inspired by score voting (also used in **LSPVRC** online voting system, Chapter 4), which enables voters to assign points to different candidates directly without any restrictions apart from the total number of available points specified (Figure 4.1).

Our proposed decentralized voting system uses the exponential ElGamal encryption system (Section 2.1.1) and an open ballot network protocol (Section 2.1.5). The additive homomorphism property of the cryptographic system makes it possible to tally encrypted ballots directly without decrypting them. This proposed system also incorporates cryptographic proofs to ensure the integrity of the voting process and to verify the validity of each ballot before it is saved to the Blockchain. To the best of our knowledge our voting system is the first decentralized ranked choice online voting system in existence, which meets the following security requirements in Chapter 1.

5.2.1 Decentralized Voting with Smart Contract

Ethereum provides a natural platform for our distributed voting system, in that it provides a decentralized “public bulletin board" to support coordination amongst voters. The execution of the election procedure is enforced by the same consensus mechanisms that secures the Blockchain. The smart contract code is stored on the Blockchain and executed by all peers to reach consensus on its output.

We present a simple voting contract written in Ethereum's Solidity language. The implementation was deployed on Ethereum's Kovan test network and the contract's interface is as follows: **VotingContract**(candidateList, voterList, definedPoint): this is the constructor function of the contract. In order for the election administrator to deploy a new contract, there are three para-

meters that have to be provided: 1) a list of candidates; 2) a list of eligible voters; and 3) total available points. Once the contract is deployed, it is immutable.

submitVote(vote, voterSign): an eligible voter is able to cast and submit a vote via this function. This function calls a contract internal **verifyPendingVote**(vote) function, which verifies the eligibility of the vote. The function returns true (success) or false.

verifyAddedVote(voterID) constant returns (bool): Each voter is able to verify the eligibility of any other voter's submission before self-tallying.

tallyVotes(candidateName) constant returns (uint8): voters can tally any candidate's final received points independently by using this self-tallying function.

The above voting contract submits and stores data in plaintext format. In order to protect the privacy of voters, an encryption system has to be used.

5.3 Our SDPVRC Online Voting System

In this section we present our proposed decentralized, self-tallying, ranked choice, smart contract-based voting system. The basic idea is as follows: The election administrator deploys a voting contract by confirming public parameters (such as the public key of the election). Each voter can then submit a ballot via the voting contract, with each ballot constituting a transaction of the Blockchain system. In case of the ballot not being verified as valid by the checks performed in the smart contract, the transaction reverts. After being mined by the Blockchain's consensus algorithm, the ballot is considered final. Figure 5.1 presents the stages of our proposed election.

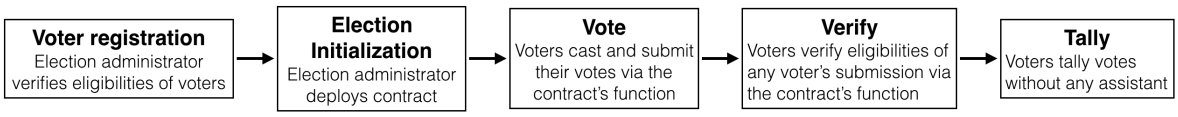

Figure 5.1: The five steps of our proposed decentralized online voting system.

The involved participants in our proposed system are:

Election administrator: An election administrator is required to set the election's parame-

ters, begin the registration stage and add voters to the list of eligible voters. The administrator should also generate a key pair (public key and private key) of the election and contribute the public key to the Blockchain. Furthermore, the administrator is responsible for voter registration, generating a candidate list, setting rules of the election, and deploying the voting contract, which cannot be changed once the election started.

Candidates: A list of candidates is generated by the election administrator. Each candidate is a contestant in the election and will receive points from different voters.

Voters: Each voter has a private key and public key. The public key is added to the Blockchain after the eligibility of the voter is verified by the election administrator. The voter can submit his/her cast ballot via the function provided by the smart contract.

Blockchain database: A distributed and append-only database. All submissions will be added to the latest block of the chain once they are verified.

The system consists of the following stages: initialization stage, registration stage, ballot casting stage, ballot verification stage and tally stage. Table 5.1 provides the notations used to explain our protocols.

Table 5.1: Notations used in the rest of Chapter 5.

n_c:	number of Candidates
n_v:	number of Voters
V_i:	i-th voter; $i \in [1, n_v]$
C_j:	j-th candidate; $j \in [1, n_c]$
$x_{v_i}^{c_j}$:	secret key of i-th voter that is used to vote for C_j candidate; $i \in [1, n_v], j \in [1, n_c]$
pk:	public key of election administrator
sk:	secret key of election administrator
P:	the pre-defined point, where the sum of all assigned points must equal to P.
$p_{v_i}^{c_j}$	a point that is assigned by V_i to C_j, $i \in [1, n_v]$, $j \in [1, n_c]$
p^{c_j}	total received point of C_j, $j \in [1, n_c]$
PZK{...}:	proof of zero knowledge

5.3.1 Initialization Stage

Before an election can start the cyclic group (G, p, g) is defined. The election administrator generates an ElGamal key pair (public key pk and secret key sk), and pk is added to the Blockchain database, which can be accessed by all voters.

The only rule for defining the election parameters is that the sum of all assigned points must be a fixed number (which we treat it as P), the election administrator defines a list of the candidates and the value of P before the election starts.

5.3.2 Registration Stage

In order to register, each voter must select n_c secret keys $x_{v_i}^{c_j} \in Z_p$ and compute the n_c corresponding public keys $g^{x_{v_i}^{c_j}} \pmod p$. The voter must register his real-world identification and his/her n_c public keys to the election administrator. Once the eligibility is verified, the voter will be added to the list of eligible voters, and all his/her $g^{x_{v_i}^{c_j}}$ will be added to the Blockchain. Once all eligible voters are registered for the election (or the deadline of registration has passed), the election administrator deploys the voting contract.

5.3.3 Ballot Casting Stage

The proposed voting system allows voters to assign different scores to different candidates according to their personal preferences. There are three phases in the voting stage: pre-computing, ballot casting and proof generation.

We do not remove the connection between the identity of voters and their ballots, meaning everyone can see that a voter submitted his/her ballot. However, the content of ballots is encrypted, meaning no-one is able to reveal the content of any individual ballot.

Pre-computing: We assume there are n_v registered voters, and all $g^{x_{v_i}^{c_j}}$ are viewable in the Blockchain database. Thus, the pre-computing values $y_{v_i}^{c_j}$ of voters can be computed by using all

other $g^{x_{v_i}^{c_j}}$ via Equation 2.3. At the end, each V_i has n_c pre-computed values as $y_{v_i}^{c_1}, y_{v_i}^{c_2}, \cdots, y_{v_i}^{c_{n_c}}$, and each value can only be used to ballot for the particular C_j.

Ballot casting: Each V_i is able to assign any integer point (from 0 to P) to different candidates, but the sum of all assigned points must equal to P (see Figure. 4.1), which is the rule each voter has to follow. Because each ballot consists of multiple assigned points (according to the number of candidates), those points are treated as private and confidential to the voters. Thus, those scores must be encrypted before submission. In our case, we use $p_{v_i}^{c_j}$ to denote a score that is assigned by voter V_i to candidate C_j, which will be encrypted twice: ElGamal encryption and distributed encryption.

ElGamal encryption Each assigned point is encrypted using the public key pk of the election administrator. For example

$$E(p_{v_i}^{c_j}, pk) = g^r,\ g^{(p_{v_i}^{c_j})} \cdot pk^r$$

meaning the score $p_{c_j}^{v_i}$ is encrypted using pk according to the ElGamal encryption.

Distributed encryption: Once the point is encrypted by ElGamal encryption, the first part (g^r) of the encrypted value will be "encrypted" again by using the private voting key (x_{v_i}) of the voter V_i, such as

$$g^r \rightarrow (y_{v_i}^{c_j})^{x_{v_i}^{c_j}} \cdot g^r$$

where $y_{v_i}^{c_j}$ is computed during the pre-computing phase and publicly accessible.

To summarise, we developed the encryption algorithm based on both the ElGamal encryption and group-based encryption, meaning each assigned point will be encrypted as per Equation 5.1.

$$E(p_{v_i}^{c_j}, pk, y_{v_i}^{c_j}, x_{v_i}^{c_j}) = (y_{v_i}^{c_j})^{x_{v_i}^{c_j}} g^r,\ g^{(p_{v_i}^{c_j})} \cdot pk^r \tag{5.1}$$

where $p_{v_i}^{c_j}$ is encrypted by using pk (public key of the election), $y_{v_i}^{c_j}$ (pre-computed value that is used by V_i to ballot C_j) and $x_{v_i}^{c_j}$ (the particular private key of V_i to ballot for C_j). Thus, a cast B_{v_i} (with n_c candidates) can be presented as:

$$B_{v_i} = \begin{bmatrix} E(p_{v_i}^{c_1}, pk, y_{v_i}^{c_1}, x_{v_i}^{c_1}) \\ \vdots \\ E(p_{v_i}^{c_{n_c}}, pk, y_{v_i}^{c_{n_c}}, x_{v_i}^{c_{n_c}}) \end{bmatrix}$$

Proof generation: In order to allow anyone to verify the eligibility of each ballot without decrypting the cipher text and revealing the content, each voter is required to generate several proofs for his/her ballot before submission (PZK denotes zero knowledge proof):

- PZK($x_{v_i}^{c_j}$): to prove each encrypted point for the candidate C_j is computed correctly using the voter's private key $x_{v_i}^{c_j}$.
- PZK(P): to prove that the sum of all encrypted points is equal to P.

The voter v_i has to generate PZK($x_{v_i}^{c_j}$) for each encrypted point $E(p_{v_i}^{c_j}, pk, y_{v_i}^{c_j}, x_{v_i}^{c_j})$, and PZK($P$) for the B_{v_i}. The summarised processing procedure of the voting stage is shown in Algorithm 7.

Remark 3 The computation details about how to generate the PZK($x_{v_i}^{c_j}$) and PZK(P) can be found in Appendix A.1.

5.3.4 Ballot Verification Stage

In order to prevent multiple counting of any individual ballot into the final result, ballot verification is required as follows:

Verify each encrypted point: In order to prevent having any error during tallying all submissions, each encrypted point $E(p_{v_i}^{c_j}, pk, y_{v_i}^{c_j}, x_{v_i}^{c_j})$ has to be confirmed as to have been computed with the correct parameters. The verification can be done by using the corresponding proofs PZK($x_{v_i}^{c_j}$) that are generated during ballot casting.

Algorithm 7: function **submitBallot**

Input : pre-defined point: P, public key pk,
all secret keys of V_i: $x_{v_i}^{c_1}, \cdots, x_{v_i}^{c_{n_c}}$
voting public keys of all voters $g^{x_{v_1}^{c_1}}, \cdots, g^{x_{v_{n_v}}^{n_c}}$

Output : B_{v_i}

computes $y_{v_i}^{c_j}$, $j \in [1, n_c]$. ▷ refer to Equation 2.3
set $B_{v_i} = []$
for $j \leftarrow 1$ **to** n_c **do**
  $E(p_{v_i}^{c_j}, pk, y_{v_i}^{c_j}, x_{v_i}^{c_j}) = \left((y_{v_i}^{c_j})^{x_{v_i}^{c_j}} \cdot g^r, g^{p_{v_i}^{c_j}} \cdot pk^r\right)$ ▷ refer to Equation 5.1
  PZK($x_{v_i}^{c_j}$): $\{K_1, K_2, Z_1, Z_2\}$ ▷ **Remark 3**
  $B_{v_i} = B_{v_i} \cup [E(p_{v_i}^{c_j}, pk, y_{v_i}^{c_j}, x_{v_i}^{c_j}), \text{PZK}(x_{v_i}^{c_j})]$
end
PZK(P): $\{T_1, T_2, s, z\}$ ▷ **Remark 3**
$B_{v_i} = B_{v_i} \cup [\text{PZK}(P)]$
$Sig_{v_i} = \text{Sign}(B_{v_i})$
if verifyPendingBallot(B_{v_i}) == False **then**
  return False
end
return $B_{v_i} = \begin{bmatrix} E(p_{v_i}^{c_1}, pk, y_{v_i}^{c_1}, x_{v_i}^{c_1}), \text{PZK}(x_{v_i}^{c_1}) \\ \vdots \\ E(p_{v_i}^{c_{nc}}, pk, y_{v_i}^{c_{nc}}, x_{v_i}^{c_{nc}}), \text{PZK}(x_{v_i}^{c_{nc}}) \\ \text{PZK}(P) \end{bmatrix}, Sig_{v_i}$

Verify sum of all encrypted points: According to the rules of the election, each voter cannot assign more than the pre-defined total available point P in his/her cast ballot. Using homomorphic addition, anyone is able to compute the sum (encrypted) of all encrypted points and verify the value by using the corresponding proof PZK(P) that are generated by the voter. The processing procedure of the verification is shown as Algorithm 8 (The purpose of function **verifyAddedBallot** is similar, but the input parameters differ).

Remark 4 The computation details about how to verify the PZK($x_{v_i}^{c_j}$) and PZK(P) can be found in Appendix A.2.

5.3.5 Tallying and Revealing Stage

Once all voters have submitted their B_{v_i} and the deadline of submission has passed, the election administrator must do the following: 1) compute the tallying result (via homomorphic addition), 2) compute their partially decrypted value and proof; 3) send partially decrypted values (including

Algorithm 8: function **verifyPendingBallot**

Input : B_{v_i}, g, all $g^{x_{v_i}^{c_j}}$, all $y_{v_i}^{c_j}$
Output: Valid or Invalid

for $j \leftarrow 1$ **to** n_c **do**
  sum $* = E(p_{v_i}^{c_j}, \cdots)$
  //verify $E(p_{v_i}^{c_j}, \cdots)$ using corresponding PZK($x_{v_i}^{c_j}$) ▷ **Remark 4**
  $E(p_{v_i}^{c_j}, \cdots) = (c_1, c_2)$ ▷ refer to Algorithm 7
  PZK($x_{v_i}^{c_j}$) = $\{K_1, K_2, Z_1, Z_2\}$ ▷ refer to Algorithm 7
  compute $c = Hash(K_1 \| K_2)$
  if $(y_{v_i}^{c_j})^{Z_1} g^{Z_2} \neq K_1 \times (c_1)^c$ *OR* $g^{Z_1} \neq K_2 \times (g^{x_{v_i}^{c_j}})^c$ **then**
    return False
  end
end
//verify sum using corresponding PZK(P) ▷ **Remark 4**
assume sum = (c_1, c_2)
PZK(P) = $\{T_1, T_2, s, z\}$ ▷ refer to Algorithm 7
compute $c = Hash(T_1 || T_2)$
if $(y_{v_i}^{c_j})^s \neq (\frac{c_1}{z})^c \cdot T_1$ *OR* $pk^s \neq (\frac{c_2}{g^P})^c \cdot T_2$ **then**
  return False
end
return True

proofs) to the Blockchain.

Each point is encrypted using our developed encryption algorithm (Equation 5.1), in which the cipher texts can be computed by homomorphic addition. In this case, we can simply multiply all B_{v_i} in the Blockchain database as shown below, where we assume there are n_v voters and n_c candidates, and all B_{v_i} have been verified as valid.

$$\prod_{i=1}^{n_v} B_{v_i} = \begin{bmatrix} \prod_{i=1}^{n_v} E(p_{v_i}^{c_1} \cdots) \\ \vdots \\ \prod_{i=1}^{n_v} E(p_{v_i}^{c_{n_c}} \cdots) \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^{n_v} (y_{v_i}^{c_1})^{x_{v_i}^{c_1}} g^{r_1},\ g^{\sum_{i=1}^{n_v} p_{v_i}^{c_1}} pk^{r_1} \\ \vdots \\ \prod_{i=1}^{n_v} (y_{v_i}^{c_{n_c}})^{x_{v_i}^{c_{n_c}}} g^{r_{n_c}},\ g^{\sum_{i=1}^{n_v} p_{v_i}^{c_{n_c}}} pk^{r_{n_c}} \end{bmatrix} \tag{5.2}$$

Due to $\prod_{i=1}^{n_v} (y_{v_i}^{c_j})^{x_{v_i}^{c_j}} = 1$ (refer to Section 2.1.4), $\prod_{i=1}^{n_v} B_{v_i}$ can be treated as n_c ciphertexts by ElGamal encryption, such as $E(\sum_{i=1}^{n_v} p_{v_i}^{c_j}), j \in [1, n_c]$.

The election administrator then has to compute partially decrypted values, such as $(g^{r_1})^{sk}, \cdots, (g^{r_{n_c}})^{sk}$.

He/she must also generate the corresponding proof for each partially decrypted value to prove that each value is computed correctly using the secret key sk. Finally, the election administrator broadcasts the partially decrypted values (including the corresponding proofs PZK(sk)) to the Blockchain. The winner of the election can be computed by any voter with Algorithm 9.

Algorithm 9: function **tallyBallots**

Input : all valid ballots $B_{v_1}, \cdots, B_{v_{n_v}}$ in Blockchain
all partial decryption values $(g^{r_1})^{sk}, \cdots, (g^{r_{n_c}})^{sk}$ by election administrator

Output : $p^{c_1}, \cdots, p^{c_{n_c}}$

compute $\prod_{i=1}^{n_v} B_{v_i}$ ▷ refer to Equation 5.2

//verify each partial decryption value using corresponding PZK(sk)

for $j \leftarrow 1$ **to** n_c **do**

  verify $(g^{r_j})^{sk}$ using corresponding PZK(sk) ▷ **Remark 5**

end

//reveal result for all candidate using partial decryption values

for $j \leftarrow 1$ **to** n_c **do**

  $p_{c_j} = \frac{\prod_{i=1}^{n_v} g^{p_{v_i}^{c_1}} pk^{r_j}}{(g^{r_j})^{sk}} = \frac{g^{\sum_{i=1}^{n_v} p_{v_i}^{c_j}} (g^{sk})^{r_j}}{(g^{r_j})^{sk}} = g^{p_{v_1}^{c_j} + \cdots + p_{v_{n_v}}^{v c_j}}$ ▷ refer to Section 2.1.4

end

return $p_{c_1}, p_{c_2}, \cdots, p_{c_{n_c}}$

Remark 5 The verification of each partial decrypted value can be treated as verifying if $(g^{r_j})^{sk}$ has the same exponentiation as pk, where $pk = g^{sk}$. The procedure is the same as the example in Section 2.1.3.

5.4 Security Analysis on SDPVRC

This section is devoted to a theoretical security analysis of our system. Noted that none of the previous related papers provided a formal security model, including only a description and an informal security discussion of their systems. Our analysis makes the following assumptions: 1) the election administrator and voters are always identifiable, as all Blockchain transactions are signed with sender's private key. 2) Voters will never disclose their private voting keys $x_{v_i}^{c_j}$; 3) the Blockchain database is secure and insert-only; 4) Our system relies on several cryptographic protocols, which are presented in Section 2 and have reliable published proofs of their security.

The security assumptions of our system are required by the previously published algorithm that our systems uses, as explained below.

Theorem 13. *If the digital signature algorithm (such as DSA) is non-falsifiable, no one is able to submit a ballot by impersonating another voter.*

Proof. In order to prevent adversaries from casting ballots by impersonating authenticated voters, we require each voter to submit with his/her digital signature algorithm. In our proposed system, only eligible voters are added to the voters list by the election administrator once their identities have been verified. The signing_verify key of each verified voter is stored on the Blockchain, and the voter is responsible for keeping their signing key secret. Once the election starts, each authorized voter signs their ballots by using his/her signing key and submits the ballot along with their signature. The smart contract is able to verify if each submission by verifying the digital signature using the corresponding signing_verify key. ■

Theorem 14. *Only one submission from each voter is accepted as valid.*

Proof. In our proposed system, only the content of a cast ballot is encrypted, the identification of the voter (and the digital signature) is in plaintext and can be viewed by everyone. Thus, multiple-voting detection is achieved by our system, as it can always detect whether a voter has previously submitted a ballot. Furthermore, depending on the requirements of the particular scenario, our system can accept one submission of each voter or accept multiple submissions for each voter and use the last ballot as valid. ■

Theorem 15. *If the underlying cryptographic systems are semantically secure, then the ballots' contents will never be revealed to anyone (including the election administrator).*

Proof. Every ballot is encrypted twice before submission. We use the ElGamal cryptosystem and a distributed encryption algorithm, which inherits the homomorphic property from the standard ElGamal system. Both algorithms are semantically secure.

All the submitted ballots remain in encrypted form as cipher texts all the time. The homomorphic property makes it possible to add all encrypted ballots without decrypting them. Furthermore, there is no relationship between the cipher texts and the corresponding plaintexts since the cryptosystem employed is probabilistic. It applies random numbers, so that the cipher text can take on different values even when the encryption is computed from the same input. Finally, due to each value being encrypted by both the public key of the election administrator and the secret voting key of the voter, the decryption must be done via collaboration of the election administrator and the voter. This means that, if the voter kept his/her secret voting keys as secret all the time (that is also one of our assumptions), even the election administrator cannot reveal anything. ■

Theorem 16. *Integrity of all cast ballots are secured after submission.*

Proof. Firstly, we require voters to sign their cast ballots by using their signing keys (refer to Algorithm 7), and we assume voters do not share their signing keys, to ensure that nobody can modify the content of a submission and fake the voter's signature. Secondly, all cast ballots will be verified being before being added to the Blockchain. Third, all verified ballots will be added to the Blockchain, being logged in an immutable ledger. Thus, the integrity of all submitted ballots is treated as secure. ■

Theorem 17. *Invalid ballots can be detected by any individual voter.*

Proof. Each cast ballot is added to the Blockchain database with corresponding proofs, generated by using Zero Knowledge Proof. The verification algorithm is public to all voters, which means the voters are able to verify any ballot without any assistant. ■

Theorem 18. *The self-tallying algorithm is proposed public accessible and anyone can use it to tally ballots without assistant.*

Proof. Once all ballots are verified and added to the Blockchain, we require the election administrator to compute the partially decrypted value in order to allow voters to compute the tallied result by themselves. In the meantime, the election administrator must generate corresponding proofs to convince all voters that all partially decrypted values are computed correctly. ■

Theorem 19. *Voters are able to verify everything of the election.*

Proof. In our system, all content (encrypted ballots, proofs and signature) for each submission is broadcasted and added to the Blockchain database, where they can be accessed by anyone. We assume that the Blockchain database is secure, and it is "append-only". The voters can do the following without any assistant:

1) Voters can verify the Blockchain transactions themselves.

2) Voters can verify the integrity of each submission by using the corresponding signing_verify keys from all voters.

3) Voters can verify each partially decrypted value (computed by election administrator) is computed correctly.

4) Voters can self-tally all ballots and compute the final result of the election. ■

5.5 Performance Analysis on SDPVRC

This section discusses the performance of our proposed voting system. The analysis is based on the computation time of each processing step, separated into 3 phases, ballot casting performance, ballots verification performance and ballots tallying performance. In our proposed protocol, each ballot is encrypted twice using different keys (common key of election administrator and secret key of the voter, refer to Algorithm 7). All tests were performed using a 512-bit key (p is 512-bit), which provides a higher security level than one-time encryption using a 1024-bit key.

We tested our proposed protocol using a high performance implementation of libgmp via the gmpy2 python module [GMP], on a laptop with the following specifications: 2.8GHz quad-core

Intel Core i7 with 6MB shared L3 cache and with 16GB of 1600MHz DDR3L on-board memory.

We use t to denote the computation time of one exponentiation, where $t = 0.09$ milliseconds. ElGamal encryption requires two exponentiations, and ElGamal decryption requires one exponentiation, where the division can be avoided by using an alternative method [WIK]. Thus, we use t_E and t_D to denote the computation time of encryption and decryption, respectively, where $t_E = 2t$ and $t_D = t$, approximately. Pre-computed values of distributed encryption (refer to Equation 2.3) require one exponentiation (the inverse power computation), and encryption also has cost of one exponentiation.

5.5.1 Performance of Ballot Casting

The performance can be analysed for the following aspects:

Total computation time: According to the Algorithm 7, we use T_{voter} to denote the total time spent before submission (including the proof generation time), where

$$T_{\text{voter}} = (t_E \times n_c + t * n_c) + (3 * n_c * t) + (3 * t) = (6n_c + 3)t$$

In this experiment, we tested T_voter in five rounds, varying the number of candidates ($n_c = 3, 5, 10, 15, 20$). The result is shown in Figure 5.2.

From the results in Figure 5.2, we can see the time cost for casting a ballot is less than 12 milliseconds even if there are 20 candidates to be ranked.

Total submission size: We assume the size of digital signature is 1024-bit (refer to Algorithm 7), and we use S_{ballot} to denote the total submission size (bits) for a voter,

$$S_{\text{ballot}} = (1024 \times n_c) + (2048 \times n_c) + (2048 + 512 + 2 * n_c * 512) + (1024) = 4096 * n_c + 3584.$$

The test result is shown in Figure 5.3 based on different numbers of candidates ($n_c = 3, 5, 10, 15, 20$).

From the result of Figure 5.3, we found the submission size of one ballot is less than 11KB even for a 20-candidate ballot.

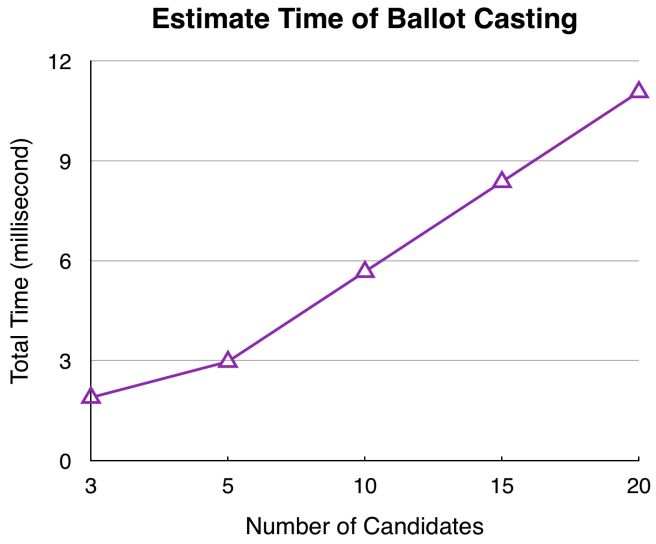

Figure 5.2: Performance of voter side when the number of candidates is 3, 5, 10, 15, 20: (a) Time spent encrypting a cast ballot, including generation time of all proofs (b) The size of a submission, includes all encrypted values and all proofs.

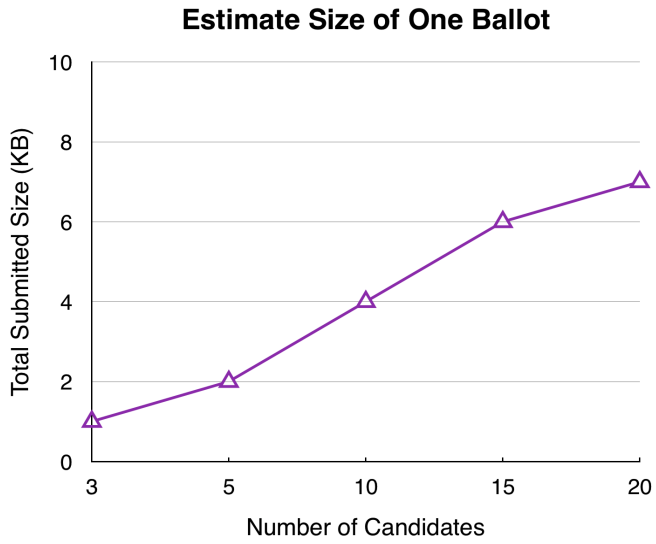

Figure 5.3: Performance of voter side when the number of candidates is 3, 5, 10, 15, 20: The size of a submission, includes all encrypted values and all proofs.

5.5.2 Performance of Ballots Verification

We have also evaluated the performance of the verification time for submissions a member of the public or an independent observer might with to verify. Due to the verification of each voter's identification being equivalent to verifying the digital signature of each submission, this is not

computationally expensive. Thus, we concentrated on the performance of Algorithm 8. We use T_{verify} to denote the total time spent verifying ballots and n to denote the total number of ballots being verified, which can be presented as follows:

$$T_{\text{verify}} = \big((5t \times n_c) + (n_c + 1 + 2 * n_c)t + (6t)\big) \times n = (8 * n_c + 7)t.$$

We tested T_{verify} in five rounds, varying the numbers of ballots verified (n = 1000, 3000, 5000, 8000, 10000). In this experiment, we assume the number of candidates is 10 ($n_c = 10$), and the result is shown in Figure 5.4.

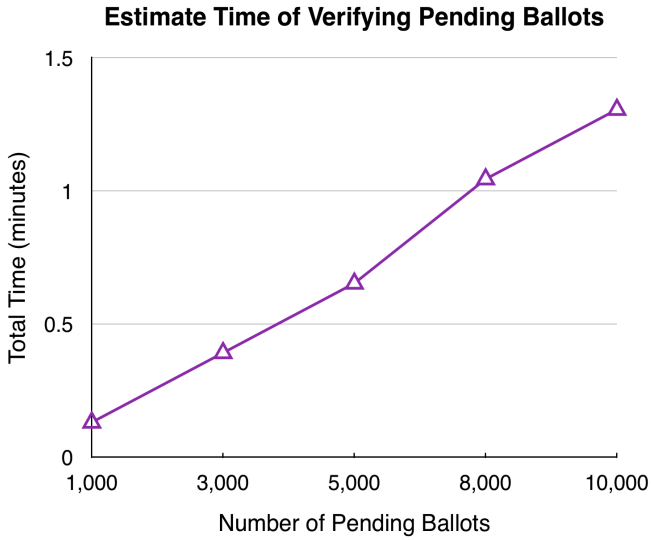

Figure 5.4: Estimate time spent of verifying 1000, 3000, 5000, 8000, 10000 ballots.

From the results in Figure 5.4, we found the time spent verifying 10,000 ballots costs less than 1.5 minutes.

5.5.3 Performance of Tallying and Revealing

Our proposed system allows voters to self-tally all submitted ballots by using all partially decrypted values from the election administrators. However, before tallying starts, each partially decrypted value must be verified using the corresponding proofs (refer to Section 5.3.5). T_{reveal}

is used to denote the total time spent verifying all partially decrypted values, tally all ballots, reveal the result (refer to Algorithm 9), which is presented as:

$$T_{\text{reveal}} = (4t * n_c) + (n_c * t) = 5 * n_c * t$$

Again, we tested T_{tally} in five rounds varying the number of candidates ($n_c = 3, 5, 10, 15, 20$). The result of this experiment is shown in Figure 5.5.

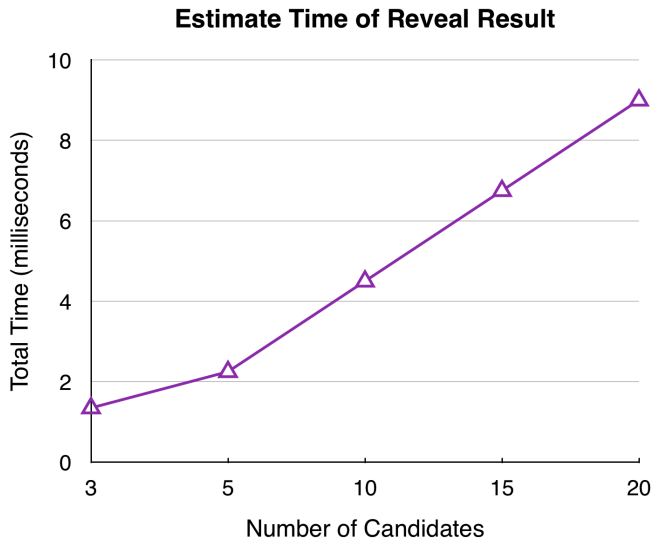

Figure 5.5: Estimate time spent of tallying all ballots, including verifying all partially decrypted values.

We found the time spent tallying all ballots (including verifying all partial decryption proofs) costs less than 10 millisecond using the same test machine.

5.6 Conclusion

In this chapter, we have proposed a **SDPVRC** (semi-decentralized public verifiable ranked choice) online voting system using cryptography and a smart contract, which allows the voters to cast their ballots by assigning arbitrary numbers of points to different candidates. This means that the voters can assign equal points to different candidates, or they can assign different points

to different candidates. The security and performance analysis confirm the feasibility of our proposed cryptographic voting contract.

To the best of our knowledge, this proposed system is the first decentralized ranked choice online voting system in existence, which can hide the content of each voter's submission. The underlying Ethereum platform enforces the correct execution of the voting protocol. We also present a security and performance analysis, showing the feasibility of our proposed protocol for real-world voting applications at large scale.

Chapter 6

A STDPVRC Online Voting System without any Tallying Authority

6.1 Motivation of STDPVRC

In the previous chapter, we proposed a **SDPVRC** (**Semi-Decentralized Public Verifiable Ranked Choice**) online voting system by using smart contract over Ethereum Blockchain. In the **SDPVRC**, there is only one tallying authority needed, and the authority can never reveal any voter's submission. However, the authority is required to compute the partial decryption value in order to allow all voters to reveal the final outcome of the election. Thus, we must assume the authority will do the correct things, otherwise, the system is disrupted, which means no one can compute the final outcome of the election.

In organising elections, a difficult problem is to achieve the trust of tallying authorities in the tallying process. Practical elections often lead to recounting of the submitted votes and raise questions about the validity of many submitted votes. There are even situations when the opposition raises concerns about the validity of the whole election process due to insufficient transparency in the verification of the votes and in tallying.

In this chapter, we aim to solve the issue, and proposed a decentralized online voting system

without any tallying authority. The present chapter proposes a **STDPVRC** (**Self-Tally Decentralized Public Verifiable Ranked Choice**) online voting system based on the Blockchain technology. The main advantages of **STDPVRC** are that all votes are submitted with complete proofs of validity and are available for public access in an encrypted form. The proposed self-tallying protocol makes the results of the election publicly computable and verifiable without any tallying authority.

6.2 Preliminaries of STDPVRC

Online voting systems that support verifiability usually assume the existence of a public bulletin board that provides a consistent view to all voters. In practice, an example of implementing the public bulletin board can be seen in the yearly elections of the International Association of Cryptologic Research (IACR) [fCR16]. It uses the Helios voting system [Adi08] whose bulletin board is implemented as a single web server. This server is trusted to provide a consistent view to all voters. Instead of such a trust assumption, [MSH17] explored the feasibility of using the Blockchain as a public bulletin board, in which the voters are responsible for coordinating communications among themselves.

There are already proposals to use a Blockchain for online voting. The Abu Dhabi Stock Exchange is launching a Blockchain voting service [Hig16] and a recent report [Bou16] by the Scientific Foresight Unit of the European Parliamentary Research Service discusses whether Blockchain-enabled e-voting will be a transformative or incremental development. In practice, companies such as The Blockchain Voting Machine [Her15], FollowMyVote [Fol16] and TIVI [Sab16] propose solutions that use the Blockchain as a ballot box to store voting data.

These solutions achieve voter privacy with the involvement of trusted third parties (e.g. a central server or tallying authorities). Under our proposed system, the voters' privacy does not need to rely on the trust in a central server, and all submitted votes can be counted without involving third parties in tallying. Our protocols allow each voter to rank all candidates by

assigning different scores to them, rather than just voting for one candidate, which is slightly different from the voting mechanism in **SDPVRC** system (**Chapter 5**).

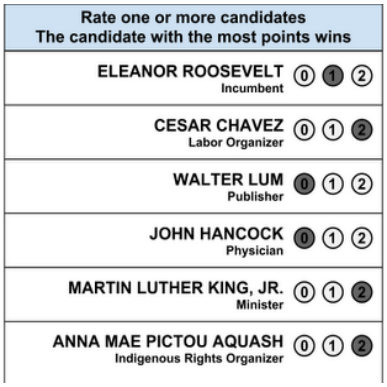

Figure 6.1: Each voter can assign different scores to the different candidates. However, the range of each score is defined by the election. In this case, the voter can only assign 0 or 1 or 2 to each candidate (cf. `https://en.wikipedia.org/wiki/Range_voting`).

6.2.1 Our new implicit verification protocol

In our implicit verification protocol, each submitted vote is protected by our new encryption mechanism, which invokes the public key of the voter and the public keys of all the candidates. When the secret keys of the candidates are released at the end of the vote, the value of each submitted vote remains protected by the public key of the corresponding voter. This is why all submitted votes remain confidential even after the verification and tallying process.

In this case, the ElGamal encryption algorithm (cf. Section 2.1.1) is used. Given a cyclic group G of a prime order q with a generator g, the secret key is sk, and the public key is $pk = g^{sk}$.

We assume that there are n different messages $m_1, m_2, \ldots, m_n$. The prover computes a ciphertext $E(m_i, pk)$, and generates proofs for a statement "$E(m_i, pk)$ is one of $E(m_1, pk), \ldots,$ $E(m_n, pk)$, where $1 \leq i \leq n$ is true. The verifiers can verify the statement without decrypting $E(m_i, pk)$, meaning they will never know which one is the truth [CDS94].

In the following example, we assume $m_i = m_1$ (all below computation included mod p at the end, for easy reading, we ignore to write them):

Prover

- generates a random number $r \in \mathbb{Z}_q$,
- computes $E(m_1, pk) = (c_1, c_2) = \{g^r, g^{m_1} \cdot pk^r\}$,
- generates random numbers $t_j \in \mathbb{Z}_q$, where $j \in [1, n]$,
- generates random numbers $v_j \in \mathbb{Z}_q$, where $j \in [2, n]$,
- computes $s_j = r \cdot v_j + t_j$, where $j \in [2, n]$,
- computes $T_{0j} = g^{t_j}$, where $j \in [1, n]$ (e.g. $T_{01} = g^{t_1}$),
- computes $T_j = (g^{m_j \cdot v_j} \cdot pk^{s_j}) / {c_2}^{v_j}$, where $j \in [1, n]$,
- $T_1 = pk^{t_1}$ in this case, because we assume $m_i = m_1$ (if $m_i = m_3$, $T_3 = pk^{t_3}$),
- computes

$$v = \text{Hash}(c_1 \| c_2 \| T_{01} \| T_{02} \| \ldots \| T_{0n} \| T_1 \| T_2 \| \ldots \| T_n)$$

- computes $v_1 = v \oplus v_2 \oplus v_3 \oplus \cdots \oplus v_n$, $\oplus$ denotes XOR,
- computes $s_1 = r \cdot v_1 + t_1$,
- sends $c_1, c_2, T_{01}, \ldots, T_{0n}, v_1, \ldots, v_n, s_1, \ldots, s_n$ to **Verifier**.

Verifier

- computes $T_j = (g^{m_j \cdot v_j} \cdot pk^{s_j}) / {c_2}^{v_j}$ since the verifier knows $g, m_j, v_j, pk, s_j, c_2$ and v_j
- verifies if

$$v_1 \oplus v_2 \oplus \cdots \oplus v_n = \text{Hash}(c_1 \| c_2 \| T_{01} \| \ldots \| T_{0n} \| T_0 \| \ldots \| T_n),$$

- verifies if $g^{s_j} = T_{0j} \cdot {c_1}^{v_j}$, where $j \in [1, n]$.

There are 4 types of proofs available: T_{0j}, T_j, v_j and s_j. The prover only sends T_{0j}, v_j and s_j.

Every verifier can 1) compute T_j using s_j and v_j; 2) verify v_j using T_{0j} and T_j; 3) verify s_j using T_{0j} and v_j. The verification processes proceed iteratively in a cycle so that, if all verification steps succeed, then this means all values in the submitted vote are correct.

The following reasons explain why the verifier cannot recover the secret information from the proofs.

1. Each T_{0j} is computed by different random number t_j, which did not send/transfer at all.

2. The equality $T_{0j} = g^{t_j}$ is satisfied. It does not reveal the value of t_j to the verifier, because the secret key is x, the public key is g^x, and x is never revealed.

3. The equality $s_j = r \cdot v_j + t_j$ holds true. The verifier knows s_j and v_j, but they don't know r and t_j. Each value t_j is used only once to compute the particular s_j.

4. Only in the "real proof" the equality $T_j = pk^{t_j}$ holds. However, for verifier, all $T_j = (g^{m_j \cdot v_j} \cdot pk^{s_j})/{c_2}^{v_j}$, which did not use t_j at all.

Thus, the verifier cannot reveal any values r, t_j and m_i from the proofs.

If all verification tests return true, it can be concluded that the statement is true. This means that then the verifiers can trust that the ciphertext is one of the form $E(m_1, pk), \ldots, E(m_n, pk)$. However, they can never know which one of the messages has been encrypted in the ciphertext.

Since all the values used to compute each T_j are public values, any verifier can compute $T_j = (g^{m_j \cdot v_j} \cdot pk^{s_j})/{c_2}^{v_j}$ by using $g, m_j, v_j, pk, s_j, c_2$.

However, for the prover, only the "real proof" is pk^{t_j}, all other T_j are "fake proofs". For the verifier, all proofs are "fake proofs", even the "real" one pk^{t_j} also equals $T_j = (g^{m_j \cdot v_j} \cdot pk^{s_j})/{c_2}^{v_j}$. Thus, there is no way to know which one is real. In other words, the prover can prove that a ciphertext $E(m_i, pk)$ is either $E(1, pk)$, $E(2, pk)$ or $E(3, pk)$ when $n = 3$, but the verifier can never figure out whether m_i is equal to 1 or 2 or 3.

6.3 Our STDPVRC Online Voting System

We propose a protocol for an online score voting with proofs of the validity of votes and tallying process, which requires all registered voters to make sure that they submit a valid vote. The encryption of the keys is organised using split secret keys shared among all the candidates or parties participating in the votes and the representatives of the organisers and the independent observers of the election.

The keys are only decrypted for those voters, who did not submit a vote. After the deadline for voting is passed, the keys of the voters who did not submit a valid vote are recovered, and the system automatically submits dummy votes with equal zero ranks for all candidates. These dummy votes do not modify the outcome of the election, but make sure that all our formulas are applicable. To ensure trust in this step, it is made possible only when all candidates and the representatives of the election organisers and observers provide their shared keys for this step to proceed.

This proposed system has common major common stages illustrated in Figure 6.2, they are initialization stage, voter registration stage, ballot casting stage, ballot verification stage, and tallying and revealing stage.

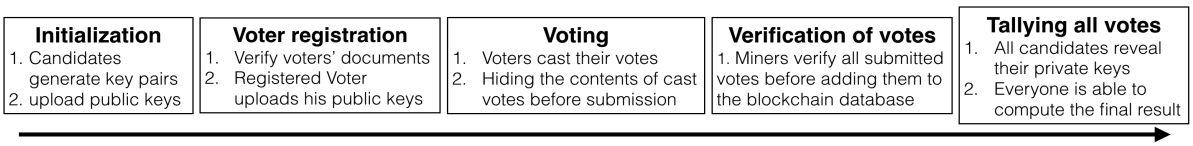

Figure 6.2: Major common steps of proposed decentralized voting system.

The system consists of the following stages: initialization stage, registration stage, ballot casting stage, ballot verification stage and tally stage. Table 6.1 provides the notations used to explain our protocols.

Table 6.1: Notations used in the rest of Chapter 6.

V_i:	i-th voter, $i \in [1, n_v]$.
n_v:	the total number of voters.
V_j:	j-th candidate, $j \in [1, n_c]$.
n_c:	the total number of candidates.
x_{v_i}:	voting private key of i-th voter, $i \in [1, n_v]$.
$g^{x_{v_i}}$:	voting public key of i-th voter, $i \in [1, n_v]$.
y_{v_i}:	pre-computed (public) value of i-th voter, $i \in [1, n_v]$.
sk_{c_j}:	private key for retrieving all votes for j-th candidate, $j \in [1, n_c]$.
pk_{c_j}:	public key of j-th candidate, $j \in [1, n_c]$.
p:	the highest score that can be assigned to any candidate.
$p_{v_i}^{c_j}$:	the score that was assigned by voter V_i to candidate C_j.
p^{c_j}:	the final tallied score of candidate C_j.
B_i:	the vote that is submitted by voter V_i.
Sig_{v_i}:	digital signature (signed by V_i's private key).
PZK(x_{v_i}):	zero knowledge proof of x_{v_i}. We use it to prove that a computed value contains x_{v_i} without revealing x_{v_i}.
PPK($p_{v_i}^{c_j}$):	partial knowledge proof of $p_{v_i}^{c_j}$. It is used to prove the range of $p_{c_j}^{i}$ without revealing $p_{c_j}^{i}$.

6.3.1 Initialization Stage

In the initialization stage, all candidates generate their key pairs (private key and public key), and broadcast them to the Blockchain database which can be accessed by anyone.

All voters and candidates agree on (G, g), where G denotes a finite cyclic group of prime order q in which the Decisional Diffie-Hellman (DDH) problem is intractable, and g is a generator in G.

We assume there are n_c candidates. Each candidate c_i chooses a private key $sk_{c_i} \in \mathbb{Z}_q$, and computes the public key $pk_{c_i} = g^{sk_{c_i}}$, where all pk_{c_i} will be broadcast to the Blockchain database, which is publicly readable but becomes immutable once the content is added to that database. Thus, the first block should contain all candidates' public keys, such as $pk_{c_1}, \ldots, pk_{c_{n_c}}$, which are generated by the election organizers.

We assume there are n_c candidates. Each candidate c_i chooses a private key $sk_{c_i} \in \mathbb{Z}_q$, and computes the public key $pk_{c_i} = g^{sk_{c_i}}$, where all pk_{c_i} will be broadcast to the Blockchain database, which is publicly readable but becomes immutable once the content is added to that database. Thus, the first block should contain all candidates' public keys, such as $pk_{c_1}, \ldots, pk_{c_{n_c}}$, which are

generated by the election organizers.

Further, the highest accepted score p is defined in this stage. For example, if $p = 3$, the score for each candidate can only be 0 or 1 or 2 or 3. Any score which is greater than p or less than 0 will result in the whole vote being treated as invalid.

6.3.2 Registration Stage

In the registration stage, all voters register to the voting system by providing verified photo identification document(s). Successfully registered voters must upload their public keys and keep their private key secret at all times.

Each voter V_i chooses a voting private key $x_{v_i} \in \mathbb{Z}_q$, and computes a voting public key $g^{x_{v_i}}$, where $i \in [1, n_v]$.

There are two ways to register to vote in the election:

(1) voters submit their identification documents (e.g. passport or driver's licence) and their voting public key for the system; and

(2) voters present their identification document at one of the voting stations in person. Both ways will require the successfully registered voters to broadcast their public voting keys ($g^{x_{v_i}}$) to the Blockchain database (publicly readable but immutable), where the second block of the database should contain all voters' voting public keys, such as $g^{x_{v_1}}, \ldots, g^{x_{vn_v}}$.

6.3.3 Ballot Casting Stage

In the ballot casting stage, each voter casts his/her ballot by assigning different scores to different candidates, and protects the contents by using his/her own private key and all candidates' public keys.

We assume all voters are registered, which means that all public voting keys $g^{x_{v_i}}$ of all successfully registered voters are added to the Blockchain database. Thus, the pre-computing value y_{v_i} of any voter can be computed by using all other $g^{x_{v_i}}$ via Equation 2.3.

Our proposed e-voting protocols allow voters to assign different scores to different candidates according to their personal preferences, and the winner is the candidate who receives the highest score. There are three phases in the voting stage: pre-computing, ballot casting and proof generation.

Our proposed voting protocols do not remove the connection between the identity of voters and their ballots, meaning everyone can see if or when a voter submitted his/her ballot. However, the content of their ballots are encrypted, meaning no-one is able to reveal the content of any individual ballot.

Each voter is able to assign any score between 0 and p to candidates. Because each vote consists of multiple scores, those scores are treated as private and confidential to the voters. Thus, the scores must be encrypted before submission. In our case, we use $p^i_{c_j}$ to denote a score that is assigned by voter V_i to candidate C_j, which is encrypted using our novel encryption mechanism presented in Section B with computation details.

Denote by $p^{c_j}_{v_i}$ the score assigned by voter V_i to candidate C_j. Applying ElGamal encryption and using the public key pk_{c_j} of candidate C_j, we get the ciphertext

$$E(p^{c_j}_{v_i}, pk_{c_j}) = g^r,\ g^{(p^{c_j}_{v_i})}(pk_{c_j})^r.$$

After that, we apply group-based encryption (cf. Section 2.1.4) to mask the first part of the ElGamal ciphertext. Namely, the first part g^r of the encrypted value is “encrypted" again by using the pre-computed value y_{v_i} explained in Section 2.1.4 and the private voting key x_{v_i} of the voter V_i, according to the mapping

$$g^r \rightarrow (y_{v_i})^{x_{v_i}} \cdot g^r.$$

This means that each score $p^i_{c_j}$ is encrypted by applying the following combined formula (6.1).

$$E(p^{c_j}_{v_i}, pk_{c_j}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} g^r,\ g^{(p^{c_j}_{v_i})} \cdot (pk_{c_j})^r. \tag{6.1}$$

where the score $p^{c_j}_{v_i}$ is encrypted by using y_{v_i} (voting public keys of all voters), pk_{c_j} (public key of the score’s recipient) and x_{v_i} (the voting private key of the voter). Finally, the whole cast vote B_i

can be presented as

$$B_i = \begin{bmatrix} E(p_{v_i}^{c_1}, pk_{c_1}, y_{v_i}, x_{v_i}) \\ \vdots \\ E(p_{v_i}^{c_{n_c}}, pk_{c_{n_c}}, y_{v_i}, x_{v_i}) \end{bmatrix} \tag{6.2}$$

where a B_i can be treated as a container of different encrypted scores $(p_{v_i}^{c_1}, ..., p_{v_i}^{c_{n_c}})$ that are assigned to different candidates $(C_1, ..., C_{n_c})$ by the voter V_i.

Proof generation: In order to allow anyone (not just miners and voters, but anyone who can access the Blockchain database) to verify each encrypted score $E(p_{v_i}^{c_j}, pk_{c_1}, y_{v_i}, x_{v_i})$ without decrypting the ciphertext and revealing the content, the voters are required to generate two kinds of proofs for each encrypted score before submission: they are PPK($p_{v_i}^{c_j}$) and PZK(x_{v_i}).

- PPK($p_{v_i}^{c_j}$): partial knowledge proof of $p_{v_i}^{c_j}$, which is used to prove the value of score $p_{v_i}^{c_j}$ is in the range between 0 and p (where p is pre-defined at the Initialization stage).

- PZK(x_{v_i}): zero knowledge proof of x_{v_i}, which is used to prove the encrypted score is computed correctly using voting private key x_{v_i} of the voter V_i.

The voter v_i has to generate both of PZK(x_{v_i}) and PPK($p_{v_i}^{c_j}$) for each score $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$. The summarised processing procedure of the voting stage is shown as Algorithm 10.

PPK($p_{v_i}^{c_j}$) and PZK(x_{v_i}) are based on the proof of zero knowledge (cf. Section 2.1.3) and the proof of partial knowledge (cf. Section 6.2.1).

In order to protect the privacy of the voters, each score is encrypted (cf. Equation 6.1) before submission. However, voters have to generate corresponding proofs to prove their votes are cast correctly by observing the following:

1. the score is between 1 and p ($p = 3$ in this case);

2. the encrypted score is computed correctly using the voter's private key x_{v_i}.

An example about the computation details can be found in the Appendix B.1.

Algorithm 10: Voting (pre-computing, vote casting and proof generation).

Input : Voter V_i, private key of V_i: x_{v_i},
the highest score for each candidate: p,
all voting public keys of voters $g^{x_{v_1}},\ldots,g^{x_{n_v}}$,
all public keys of candidates $pk_{c_1},\ldots,pk_{c_{n_c}}$.

Output : B_i and Sig_{v_i}

Compute y_{v_i}. ▷ cf. Equation 2.3
// Verify if V_i submitted a B_i previously.
// Verify if the voter is eligible (all public keys of the eligible voters are added to the Blockchain).
if *search*(Sig_{v_i}) == *true* **then**
    throw; ▷ if there is a signature of v_i in the Blockchain
end
Set $\mathrm{B}_i = []$.
for $j \leftarrow 1$ **to** n_c **do**
    $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i}) = \left((y_{v_i})^{x_{v_i}} \cdot g^r, g^{p_{v_i}^{c_j}} \cdot pk_{c_j}^r\right)$. ▷ cf. Equation (6.1)
    $\mathrm{PPK}(p_{v_i}^{c_j})$: $\{T_{01},\ldots,T_{0p},v_1,\ldots,v_p,s_1,\ldots,s_p\}$. ▷ cf. Section 6.2.1
    $\mathrm{PZK}(x_{v_i})$: $\{K_1,K_2,Z_1,Z_2\}$. ▷ cf. Section 2.1.3
    $\mathrm{B}_i = \mathrm{B}_i \cup [E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i}), \mathrm{PPK}(p_{v_i}^{c_j}), \mathrm{PZK}(x_{v_i})]$.
end
Generate the signature $Sig_{v_i} = \mathrm{Sign}(\mathrm{B}_i)$. ▷ Sign the Vote using v_i's private key.
return $\mathrm{B}_i = \begin{bmatrix} E(p_{c_1}^i, pk_{c_1}, y_{v_i}, x_{v_i}), \mathrm{PPK}(p_{c_1}^i), \mathrm{PZK}(x_{v_i}) \\ \vdots \\ E(p_{c_{n_c}}^i, pk_{c_{n_c}}, y_{v_i}, x_{v_i}), \mathrm{PPK}(p_{c_{n_c}}^i), \mathrm{PZK}(x_{v_i}) \end{bmatrix}, Sig_{v_i}$.

6.3.4 Ballot Verification Stage

In the verification stage, the eligibility of each submission has to be verified before adding it to the Blockchain database, which actions are processed by miners (voters can also apply to act as miners).

The verification algorithm employed in our protocols is publicly accessible. This means that anyone with access to the Blockchain database can verify any submitted vote. Each B_i consists of multiple encrypted scores $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$, and each encrypted score has two proofs: $\mathrm{PPK}(p_{v_i}^{c_j})$, $\mathrm{PZK}(x_{v_i})$. Both $\mathrm{PPK}(p_{v_i}^{c_j})$ and $\mathrm{PZK}(x_{v_i})$ can be verified without revealing the content of the encrypted score. The processing procedure of verification is shown in Algorithm 11.

Only if both $\mathrm{PPK}(p_{v_i}^{c_j})$ and $\mathrm{PZK}(x_{v_i})$ are verified, the score $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$ can be considered as valid. And only when all encrypted scores in a B_i are verified as valid, can the B_i be treated as a verified submission, and added to the Blockchain database. An example of computation

Algorithm 11: Verify a submitted B_i (can be performed by miners or voters).

Input : a vote B_i and the public voting key of V_i: $g^{x_{v_i}}$.
Output: Accept or Reject.

for $j \leftarrow 1$ **to** n_c **do**
- $E(p_{v_i}^{c_j}, pk_{c_j}) = (c_1, c_2) = ((y_{v_i})^{x_{v_i}} \cdot g^r,\ g^{p_{v_i}^{c_j}} \cdot pk_{c_j}^r)$ ▷ cf. Algorithm 10
- $\text{PPK}(p_{v_i}^{c_j}) = \{T_{01}, \ldots, T_{0p}, v_1, \ldots, v_p, s_1, \ldots, s_p\}$ ▷ cf. Algorithm 10
- Verify $\text{PPK}(p_{v_i}^{c_j})$: ▷ cf. Section 6.2.1
- **for** $k \leftarrow 1$ **to** p **do**
  - Verify if $g^{s_k} = T_{0k} \cdot {c_1}^{v_j}$.
  - //By using publicity available values in the Blockchain
  - compute $T_k = (g^{k \cdot v_k} \cdot pk^{s_k}) / {c_2}^{v_k}$
- **end**
- Verify if $v_1 \oplus v_2 \oplus \cdots \oplus v_p = \text{Hash}(c_1 \| c_2 \| T_{01} \| \ldots \| T_{0p} \| T_1 \| \ldots \| T_p)$.
- $\text{PZK}(x_{v_i}) = \{K_1, K_2, Z_1, Z_2\}$ ▷ cf. Algorithm 10
- Verify $\text{PZK}(x_{v_i})$: ▷ cf. Section 2.1.3
- Verify if $(y_{v_i})^{Z_1} g^{Z_2} = (c_1)^{\text{Hash}(K_1 \| K_2)} K_1$.
- Verify if $g^{Z_1} = (g^{x_{v_i}})^{\text{Hash}(K_1 \| K_2)} K_2$.

end
if *all the above verifications are passed* **then**
- **return** Accept

else
- **return** Reject

details can be found in Appendix B.2.

6.3.5 Tallying and Revealing Stage

In the tallying stage, all candidates reveal their private keys, and all users who can access the Blockchain database are able to compute the final result.

Once the deadline of the election is expired. The confidentiality authority will check if all voters have submitted their votes and if they submitted valid votes. In this case, we can assume all voters have submitted their votes, all candidates must broadcast their private key (sk_{c_1}, sk_{c_2},..., $sk_{c_{n_c}}$) to the Blockchain.

Under our proposed algorithm, each score is encrypted using our developed ElGamal encryption (cf. Equation 6.1), where the ciphertexts can be computed according to homomorphic addition (cf. Section 2.1.1.1). In this case, we can simply multiply all valid B_i in the Blockchain database,

such as below.

$$\prod_{i=1}^{n_v} \mathrm{B}_i = \begin{bmatrix} \prod_{i=1}^{n_v} E(p_{v_i}^{c_1}, \ldots) \\ \vdots \\ \prod_{i=1}^{n_v} E(p_{v_i}^{c_{n_c}}, \ldots) \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} g^{r_1}, \prod_{i=1}^{n_v} g^{(p_{v_i}^{c_1})} (pk_{c_1})^{r_1} \\ \vdots \\ \prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} g^{r_{n_c}}, \prod_{i=1}^{n_v} g^{(p_{v_i}^{c_{n_c}})} (pk_{c_{n_c}})^{r_{n_c}} \end{bmatrix},$$

where we assume there are n_v voters and n_c candidates.

Since $\prod_{i=1}^{n_v} (y_{v_i})^{x_{v_i}} = 1$ (as explained in Section 2.1.4),

$$\prod_{i=1}^{n_v} \mathrm{B}_i = \begin{bmatrix} \prod_{i=1}^{n_v} g^{r_1}, \prod_{i=1}^{n_v} g^{(p_{v_i}^{c_1})} (pk_{c_1})^{r_1} \\ \vdots \\ \prod_{i=1}^{n_v} g^{r_{n_c}}, \prod_{i=1}^{n_v} g^{(p_{v_i}^{c_{n_c}})} (pk_{c_{n_c}})^{r_{n_c}} \end{bmatrix} = \begin{bmatrix} (g^{r_1}, g^{\sum_{i=1}^{n_v} p_{v_i}^{c_1}} (pk_{c_1})^{r_1}) \\ \vdots \\ (g^{r_{n_c}}, g^{\sum_{i=1}^{n_v} p_{v_i}^{c_{n_c}}} (pk_{c_{n_c}})^{r_{n_c}}) \end{bmatrix}.$$

Finally, the total score of each candidate C_j can be revealed by using the published private key sk_{c_j} of all candidates,

$$\begin{bmatrix} p^{c_1} \\ \vdots \\ p^{c_{n_c}} \end{bmatrix} = \begin{bmatrix} D\big(g^{r_1}, g^{\sum_{i=1}^{n_v} p_{v_i}^{c_1}} (pk_{c_1})^{r_1}, sk_{c_1}\big) \\ \vdots \\ D\big(g^{r_{n_c}}, g^{\sum_{i=1}^{n_v} p_{v_i}^{c_{n_c}}} (pk_{c_{n_c}})^{r_{n_c}}, sk_{c_{n_c}}\big) \end{bmatrix} = \begin{bmatrix} g^{\sum_{i=1}^{n_v} p_{v_i}^{c_1}} \\ \vdots \\ g^{\sum_{i=1}^{n_v} p_{v_i}^{c_{n_c}}} \end{bmatrix},$$

where we use p^{c_j} to denote the total tallied score of the candidate C_j, and $D(\ldots)$ denotes decryption algorithm of ElGamal decryption (cf. Section 2.1.1). The winner of the election can be determined by comparing $p^{c_1}, \ldots, p^{c_{n_c}}$. The candidate who receives the highest score is the winner.

The summarized self-tally algorithm is shown as Algorithm 12, which can be used by anyone.

6.4 Security Analysis on STDPVRC

The security level of our proposed protocols can be analysed according to the security requirements (cf. Section 1), and the analysis assumes that all voters will never share their private keys with others, and that candidates will never share their private keys with others until the voting has closed.

The identification of voters has to be verified in order to participate in the election.

Algorithm 12: Self-tally all votes

Input : all votes $B_1, \ldots, B_{n_v}$
all privates keys of candidates $sk_{c_1}, \ldots, sk_{c_{n_c}}$
Output : $p^{c_1}, \ldots, p^{c_{n_c}}$

sets $p^{c_1}, \ldots, p^{c_{n_c}} = 1$
for $i \leftarrow 1$ **to** n_v **do**
  for $j \leftarrow 1$ **to** n_c **do**
    $p^{c_j} = p^{c_j} \times E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$ ▷ cf. Section 2.1.1
  end
end
for $j \leftarrow 1$ **to** n_c **do**
  $p^{c_j} = D(p^{c_j}, sk_{c_j}) = D\left(\prod_{i=1}^{n_v} E(p_{v_i}^{c_j}, pk^{c_j}), sk_{c_j}\right) = g^{\sum_{i=1}^{n_v} p_{v_i}^{c_j}}$
end
return $p^{c_1}, p^{c_2}, \ldots, p^{c_{n_c}}$

Theorem 20. *Only successfully registered voters are able to submit their votes.*

Proof. In our protocols, all voters have to register before the election starts, and all successfully registered voters are required to upload their voting public keys, which will be added to the Blockchain database, and can be seen by everyone.

Once the election starts, each voter should generate a digital signature by using their voting private keys, and submit it along with their votes. After submission, anyone can verify the eligibility of each vote by verifying the digital signature using the corresponding voting public key.

If no corresponding key can be found in the database, or if the digital signature is found to be invalid, the submission will be treated as invalid and be discarded. ■

In order to ensure the fairness of the election, each verified voter is only allowed to contribute one vote to the election.

Theorem 21. *Each voter can only submit once, meaning multiple submissions from the same voter can be detected and discarded.*

Proof. In our protocols, the id/name of each submission is public information for everyone: only the content of the submission is encoded. Thus, multiple submissions can be easily detected by

searching the database for the id/name. If a submission has been added to the database with the same id/name, the latest submission will be discarded. Otherwise, the latest submission will be added to the database. ■

Each submitted vote contains the voting preference of the voter, which can be treated as private to the voter and must be kept secret at all times.

Theorem 22. *The content of any individual vote will never be revealed.*

Proof. In our protocols, each score is given by the expression $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$ (cf. Algorithm 10), which is encoded by using not only the public key (pk_{c_j}) of the candidate, but also the voting private key x_{v_i} of the voter.

Each encrypted score $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i})$ can be decrypted only by using both the private key of the candidate (sk_{c_j}) and the x_{v_i} of the voter. Although all candidates will reveal their sk_{c_j} at the close of voting, the $E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i})$ cannot be decrypted without the x_{v_i}.

Thus, it can be assumed the contents of any encrypted votes will never be revealed on the basis that voters will never disclose their voting private key x_{v_i} to others, and there is no way to decode the encrypted scores without those voting private keys. ■

Our protocols use a Blockchain database, which is maintained by all users (miners). All votes will be verified before being added to the database, whereupon the content cannot be further modified. In this case, we assume most users (miners) are honest, and the Blockchain database is secure and unbreakable.

Theorem 23. *No one can modify any data in a submission once it is confirmed and added to the Blockchain database.*

Proof. A Blockchain database is a decentralized database, which means there is not a centralized server managing the database, but everyone has the latest version of the whole database. In this situation, no one can actually modify any data in the database because the modification will not

be accepted by most users (miners) who are (at least generally) assumed to be honest. To sum up, no one is able to change anything from the Blockchain database without a general notification, and most users (miners) will reject any unauthorized modification, given the assumption (i.e. that most users are honest), meaning the integrity of all or any submissions cannot be compromised once they have been confirmed and added to the Blockchain database. ■

We assume everything that has been added to the Blockchain database can be accessed by everyone but is immutable, and the correctness of the final result can be analysed from two aspects.

Theorem 24. *Any individual submission can be verified by anyone without accessing the content of the submission.*

Proof. A submission can only be included in the final result if it has been verified as valid using Algorithm 11. The corresponding proofs of each submitted vote are generated by the voter according to the proof of partial knowledge (cf. Section 6.2.1) and the proof of zero knowledge (cf. Section 2.1.3), which can be verified without knowing any relevant information about the content. ■

Theorem 25. *The final tallied result can be computed by any voter (or miner) using his/her voting private key and all candidates' private keys.*

Proof. In our protocols, all candidates have to reveal their private keys once all votes are received and verified as valid. At that point, everyone or anyone can compute the final score for any candidate via Algorithm 12, which uses the private key of the particular candidate and his/her own voting private key. ■

End-to-end voter verification is achieved by our proposed protocols: the analysis has the following three aspects.

Theorem 26. *Each vote is cast-as-intended, meaning the voter is able to verify that the vote has been generated correctly as intended.*

Proof. In our protocols, each vote is cast based on ElGamal encryption (cf. Section 2.1.1) and group-based encryption (cf. Section 2.1.4) using all candidates' public keys, all voters' voting public keys and the voter's voting private key (cf. Section 10). As all the necessary elements/components are public for the voter himself/herself, he/she can re-generate the vote in order to verify that the vote has been computed correctly. ■

Theorem 27. *Each submission is recorded-as-cast, meaning any modification to the submission can be discovered without difficulty.*

Proof. ElGamal encryption is a probabilistic encryption, meaning the encrypted result will always be different even if one encrypts the same plaintext multiple times. Thus, the voter is able to save the cast vote (encrypted) as an original receipt before submission. Once the submission is added to the Blockchain, the voter can easily verify if the recorded submission is the one he/she submitted by comparing the content with the content of the original receipt. ■

Theorem 28. *Each vote is counted-as-recorded, meaning the voter is able to verify if his/her vote has contributed correctly to the final result.*

Proof. In our protocols, a self-tally algorithm (cf. Algorithm 12) can be used by any individual voter, which means everyone and any one is able to tally all submissions, find the winner without any collaboration with others, and verify that the published result is correct. ■

Theorem 29. *There is no way for a voter to prove how his/her vote is cast after submission, unless the voter's private key is revealed.*

Proof. In our protocols, each vote is encrypted by using y_{v_i}, x_{v_i} and pk_{c_i} and equation (6.1), where the x_{v_i} is the private key of the voter. The voter has to reveal all the 3 factors in order to

prove how he/she voted. However, one of our assumptions is that the voters have to keep their private keys secret all the time. Thus, there is no way to prove how the vote was cast without revealing the private key x_{v_i}. ■

6.5 Performance Analysis on STDPVRC

The analysis of performance has two subsections: client (voters) and the server (miners). All tests were performed on a laptop with the following specifications: CPU: 2.2 GHz Intel Core i7, Memory: 16 GB 1600 MHz DDR3.

All tests used high performance code from libgmp via the gmpy2 python module [GMP] with 1024-bit length key (p and q are 1024-bit). We use t to denote the computation time of one exponentiation (such as a $g^a \bmod b$), in this case, $t = 0.00012$ seconds on the laptop.

ElGamal encryption requires two exponentiations, and ElGamal decryption requires one exponentiation (cf. Section 2.1.1). The division can be avoided by calculating ${c_1}^{q-sk}$ instead of ${c_1}^{sk}$, where ${c_1}^{q-sk}$ is the inverse of ${c_1}^{sk}$. In this case, we use t_E and t_D to denote the computation time of encryption and decryption, respectively, where $t_E = 2t$ and $t_D = t$, approximately.

6.5.1 Performance on voters' side

On the voters' side, each voter casts and submits a vote to the Blockchain database. To prevent the voting preferences of each vote being revealed, and also to protect the voters' privacy, we require voters to encrypt their votes before broadcasting them to the Blockchain database.

Thus, the performance on the voters' side can be summarized as being the total computation time for casting a vote and the total submission size of the vote.

Total computation time for casting one vote. In this case, we use T_{voter} to denote the total computation time on the voters' side, where we only focus on the computation cost of all exponentiations. According to Algorithm 10, the computation time for each candidate costs an ElGamal encryption computation ($2t$, cf. Section 2.1.1), a PPK computation ($5pt - 2t$, cf.

Section 6.2.1) and a PZK computation ($3t$, cf. Section 2.1.3). Therefore, the total computation time on the voters' side can be presented as

$$T_{\text{voter}} = (5pt + 3t) * n_c$$

where p and n_c denotes the number of possible scores for each candidate and the number of candidates, respectively.

In this experiment, we tested the T_{voter} over five rounds on the laptop, according to different numbers of candidates (n_c = 3, 5, 10, 15 and 20) and different total available points ($p = n_c$). The result is shown in Fig. 6.3.

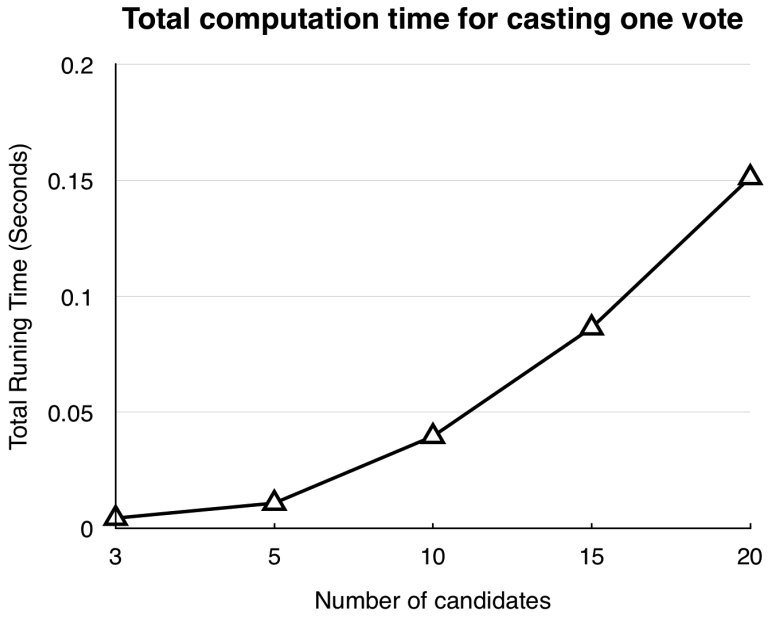

Figure 6.3: Estimate of total time spent casting one vote when the number of candidates n_c are 3, 5, 10, 15 and 20, and the corresponding p is also 3, 5, 10, 15 and 20.

From the results in Fig. 6.3, we can see that the time taken for casting one vote is less than 0.25 seconds even if there are 20 candidates in the election.

Total submission size for one vote. In this case, we use S_{one_ballot} to denote the total submission size for one vote, which contains n_c encrypted value, n_c proofs of PPK and n_c proofs of PZK. Thus, the total submission size $S_{\text{one\_ballot}}$ can be presented as

$$S_{\text{one\_ballot}} = 1024 * (3p + 6) * n_c$$

where an encrypted score for each candidate contains an ElGamal encrypted value which is $1024 * 2$ bits (cf. Section 2.1.1), a proof of PPK is $1024 * (3p)$ bits (cf. Section 6.2.1) and a proof of PZK is $1024 * 4$ bits (cf. Section 2.1.3).

In this experiment, we also tested the S_{one_ballot} over five rounds on the laptop, where the numbers of candidates (n_c) and the total available points (p) are the same as the previous experiment, which is $n_c = p = 3$, 5, 10, 15 and 20. The result is shown in Fig. 6.4.

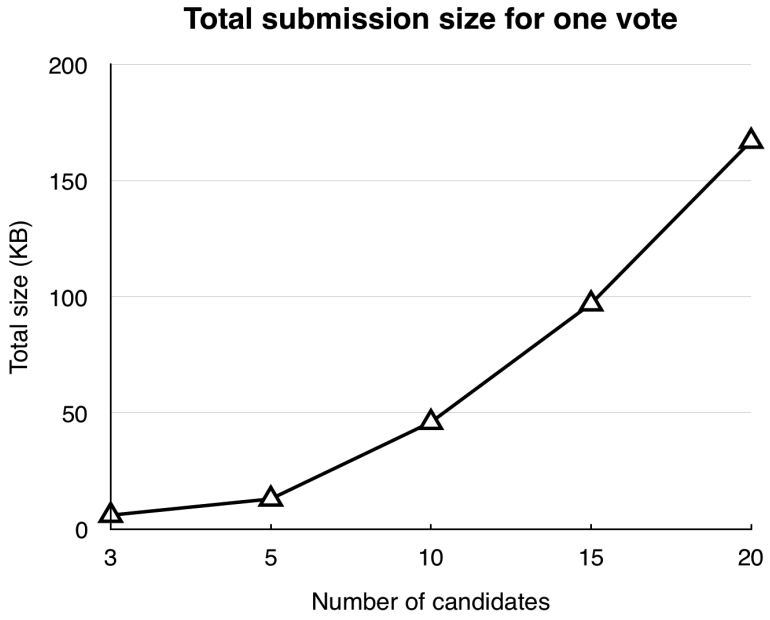

Figure 6.4: Estimate of total size of one vote when the number of candidates n_c are 3, 5, 10, 15 and 20, and the corresponding p is also 3, 5, 10, 15 and 20.

From the result of Figure 6.4, we found the size of one submission is less than 200KB even for a 20-candidate vote. Based on the publicly-available assessment of the speedtest ranks Internet access speed in more than 100 countries [Mur17], the slowest Internet speed is 3.03 Mbps and the fastest is 62.59 Mbps. It shows that the submission size for a vote (including all encrypted values and all proofs) of our protocols should not be problematic.

6.5.2 Performance on miners' side

Our protocols use a Blockchain database. Therefore, the performance on the miners' side can be summarized as having the following attributes: 1) the computation time for verifying all pending submissions and 2) the computation of self-tallying all verified votes.

Please note, here we do not discuss the computation time for solving the mathematical puzzle of the Blockchain database in order to confirm a new block: the difficulty level can be defined by the organizers.

The computation time for verifying all pending submissions. In this case, we use T_{verify} to denote the total computation time for verifying all pending submissions, which can be treated as the sum of verification time of PPK proofs and verification time of PZK proofs. According to Algorithm 11, the computation time for verifying a PPK and a PZP are $5pt$ (cf. Section 6.2.1) and $5t$ (cf. Section 2.1.3) respectively. Thus the total computation time for verifying all pending submissions is

$$T_{\text{verify}} = (5pt + 5t) * n_c * n_v$$

where n_v denotes the number of the pending submissions.

In this experiment, we assume there are 10 candidates ($n_c = p = 10$) in the election, and test the T_{verify} over five rounds on the laptop, where the numbers of pending submissions are 3, 5, 10, 15 and 20. The results are shown in Fig. 6.5.

From the results in Fig. 6.5, we found the time spent in verifying 1,000 votes takes approximately one minute using a standard laptop (the specifications of which were provided at the start of this section), where the miners usually use at least a server-power computer, which means that the running time will be further reduced.

The cost of self-tallying all verified votes. In our protocols, all voters are able to download all votes from the Blockchain database. We assume all votes must be verified before being added to this database. This means normal users do not need to download all corresponding proofs unless they want to independently verify them. Self-tallying the votes is done using Algorithm 12.

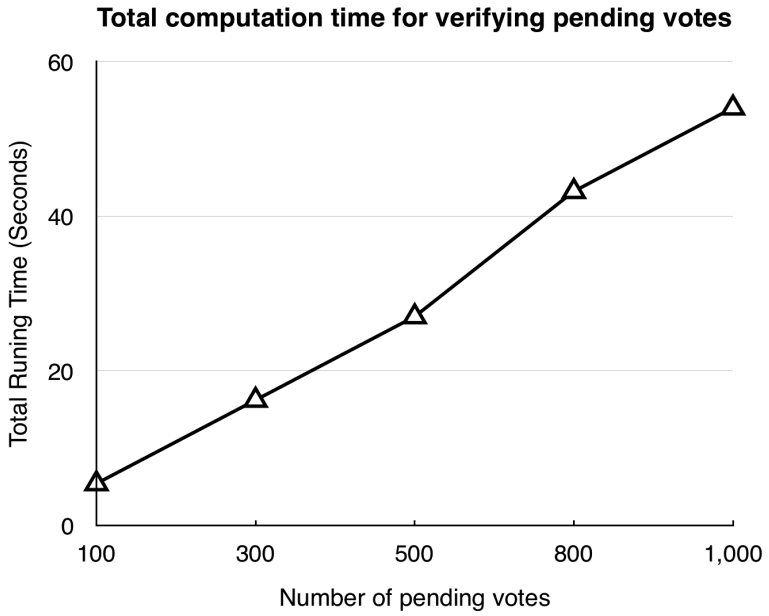

Figure 6.5: Estimate of total computation time (by using a laptop) for verifying 100, 300, 500, 800 and 1,000 pending votes, for 10 candidates in the election.

In this case, we use $S_{\text{all\_ballots}}$ and T_{tally} to denote the total size of the whole Blockchain database (excluding all corresponding proofs) and the computation time for tallying all verified votes, which can be presented as

$$S_{\text{all\_ballots}} = 1024 * 2 * n_c * n_v \text{ and } T_{\text{tally}} = t * n_c,$$

respectively, according to Algorithm 12.

In this experiment, we again assume there are 10 candidates ($n_c = p = 10$) in the election, which means $T_{\text{tally}} = t * n_c = 10t$, and we test $S_{\text{all\_ballots}}$ over five rounds on the same laptop, based on five different numbers of votes (10,000, 30,000, 50,000, 80,000 and 100,000). The results are shown in Fig. 6.6.

From the results in Figure 6.6, we found the total size of the whole Blockchain database (excluding all corresponding proofs) is less than 250 MB when there are 100,000 votes. The computation time for self-tallying all votes only regards the number of candidates: in this case, it only took 0.0012 seconds to tally them by using a standard laptop (the specifications of which were provided at the start of this section).

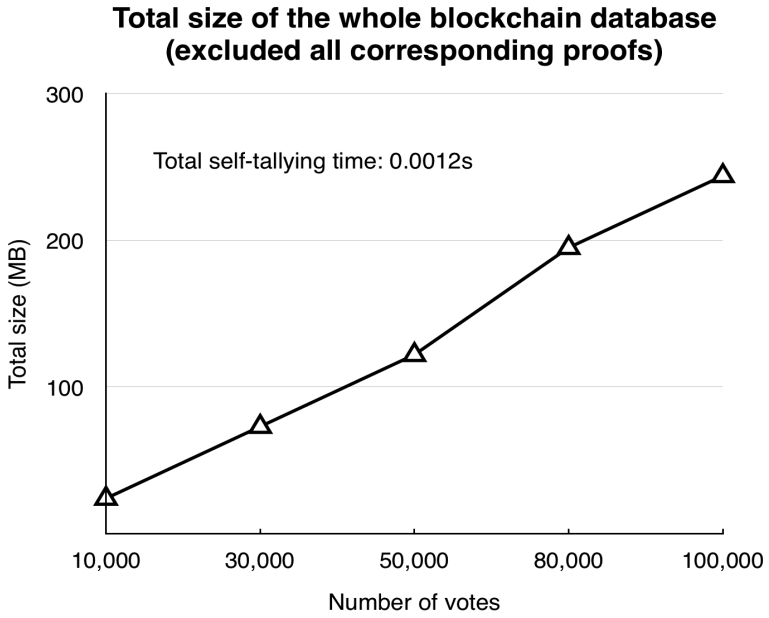

Figure 6.6: Estimate of total size of the Blockchain database (excluding all corresponding proofs) when the total number of pending votes are 10,000, 30,000, 50,000, 80,000 and 100,000 pending votes, for 10 candidates in the election.

6.6 Conclusion

In this chapter, we have proposed **STDPVRC** online voting system inspired by the **SDPVRC** system in **Chapter 5**, which is a decentralised publicly verifiable online voting protocol based on Blockchain technology and a new encryption mechanism. They store all submitted votes in a Blockchain database, which can be accessed by all users but is immutable. Our proposed protocols also allow voters to cast their ballots by assigning different points to different candidates. Each vote is encrypted before submission and remains encrypted at all times. The additive homomorphic property of the exponential ElGamal cryptosystem enables effective processing of the ciphertexts during these procedures. Moreover, the eligibility of voters and their submissions can be verified by anyone without revealing the contents of the votes, and our proposed verification and self-tallying algorithms allow any voter to verify the correctness of the final result. The whole Blockchain based voting system is processed by all users (voters and registration authorities), and tallying procedure can be processed without a need to involve a third party (e.g. tallying

authority) in computation and verification. The proposed system also provides a security and performance analysis and confirms the feasibility of our proposed Blockchain online voting protocols for real-life elections.

Chapter 7

Conclusion and Future Work

7.1 Conclusion of the thesis

The aim of the research was to build a decentralized publicity verifiable ranked choice online voting system, which not only has no tallying authority at all, but also can achieve the advanced E2E voter verifiability. This thesis also shows the processing steps of how such a system is proposed. In this thesis, we propose several public verifiable ranked choice online voting systems, which address our research questions.

The first research question is treated in Chapter 3. The main objective of this chapter is to design an easy-use ranked choice online voting system, which also achieved standard E2E verifiability. In order to protect the confidentiality of the votes, each cast ballot is encrypted before submission. Furthermore, during voting the system ensures that proofs are generated and stored for each element in the cast ballot. These proofs can then be used to verify the correctness and the eligibility of each ballot before counting without decrypting and accessing the content of the ballot. This validates the votes in the counting process and at the same time maintains confidentiality.

Regarding the second research question, an advanced ranked choice voting mechanism is designed and proposed in Chapter 4. Since the privacy of voters is protected by encrypting the

cast ballots before submission, it follows that the size of each encrypted ballot is critical for proposing a large-scale online voting system. In this chapter, the addition property of ElGamal is redesigned, so that the cryptosystem can also compute multiplication directly on the ciphertext.

For most of existing online voting systems, there is a centralized server with trusted authorities, such as tallying authorities. Once all ballots are submitted by all voters, the tallying authorities are responsible to (1) verify all submissions, (2) count only valid ballot, and (3) publish the final outcome of the election. The security level of such centralized systems is relied to whether the authorities are honest or not, because the encryption key that is used to encrypt each cast ballot is computed by all authorities. That's why there are multiple tallying authorities in the system (refer to Chapter 3 and Chapter 4). We have to assume that at least one of them is honest, since the voters' privacy will be revealed if all authorities are colluded. Under this assumption, we also require each authority to prove what and how he/she did in every step he/she is involved. Under this assumption, we also require each authority to prove what and how he/she did in every step he/she involved. However, in the real-life, it's quite difficult to persuade voters to trust the authorities in the system.

In order to reduce the dependence on the centralized server and authorities, the third research question is addressed in Chapter 5. In this chapter, we propose a semi-decentralized online voting system. In the system, each cast ballot is still encrypted before submission, but the encryption key is not just from the authorities. Each cast ballot is also encrypted by using the voter's private key. This is why the system needs only one authority, and he/she can never decrypt any voter's submission without the voter's private key. The reason why we called it as "semi-decentralized" system is the system still need one authority to help to reveal the final outcome of the election, but he/she cannot reveal any voter's privacy. Thus, the voters will never worry that their voting preferences can be revealed by any tallying authority.

However, if the only tallying authority does not help to reveal the final outcome, the system is aborted. Although the privacy of voters is secure, it's meaningless if no one can reveal the final

outcome of the election.

Finally, in Chapter 6 we propose a solution to the fourth research question. We propose further development of the protocols to make the system totally decentralized and eliminate any need to use any tallying authority. The outcome of the election can be revealed by all voters and candidates in the election. Our proposed self-tallying protocol allows the outcome can be computed by any individual without any other's assistance, which enhanced the trustworthiness of the system and vote of confidential.

Overall, a decentralized ranked choice online voting system is proposed by using Blockchain technology. The system has no hardwired restrictions on possible vote assignments to candidates, protects voter confidentiality by using a homomorphic encryption system and stores proofs for each element of a vote. Furthermore, the system achieves advanced E2E verifiability, which mostly enhances the trust in the system and the confidentiality of the voters. We propose a self-tallying online voting system without any tallying authority. The voters will never need to assume the centralized server and authorities are not compromised, because there is no centralized party at all. The candidates and voters of the election take the responsibility for the whole procedure of the election.

7.2 Future Work

In order to enhance the trustworthiness of an online voting system, we designed and proposed several systems from centralized to decentralized, which reduced the dependence of the tallying authorities step by step. However, for a decentralized online voting system without any tallying authority, there are still several issues exist, such as abortive issue by voters. In such a decentralized online voting system, we have to assume all involved voters submitted their ballots. However, in practical life, we can never make this assumption. In the future work, we are planning to address this issue.

Furthermore, we will investigate the feasibility of running a large-scale election over the

Blockchain. Based on the knowledge gained from this Thesis, we believe that if such a perspective is ever considered possible, its implementation will almost certainly require a dedicated Blockchain. For example, we build our own Ethereum-like Blockchain, which only stores the voting smart contract. This new Blockchain can have a larger block size to store more transactions on-chain and may be maintained in a centralised manner.

Appendix A

Computation details of proof generations and verifications

In order to protect the privacy of the voters, each assigned point is encrypted (refer to Equation 5.1) before submission. However, voters have to generate corresponding proofs to prove their votes are cast correctly by observing the following: 1. each encrypted score is computed correctly using the voter's private key $x_{v_i}^{c_j}$; 2. The sum of all encrypted points must be equal to the pre-defined total number of available points P.

A.1 Each encrypted value is computed correctly

We present the proofs generation and verification for an encrypted score in B_{v_1}, where we assume V_1 assigned 5 points to C_1, and the encrypted value should be $E(5, pk, y_{v_1}^{c_1}, x_{v_1}) = (c_1, c_2)$, where $c_1 = (y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot g^r$, $c_2 = g^5 \cdot pk^r$:

Prover

- generates a random number $k_1, k_2 \in \mathbb{Z}_q$
- computes $K_1 = (y_{v_1}^{c_1})^{k_1} \cdot g^{k_2}$

- computes $K_2 = g^{k_1}$
- computes $c = Hash(K_1 \| K_2)$
- computes $Z_1 = x_{v_1}^{c_1} c + k_1$
- computes $Z_2 = rc + k_2$
- sends PZK($x_{v_i}^{c_1}$) = $\{K_1, K_2, Z_1, Z_2\}$ to **Verifier**

Verifier

- compute $c = Hash(K_1 \| K_2)$
- verify if $(y_{v_1}^{c_1})^{Z_1} g^{Z_2} = K_1 \times (c_1)^c$
- verify if $g^{Z_1} = K_2 \times (g^{x_{v_i}^{c_1}})^c$

where $y_{v_i}^{c_1}$ and $g^{x_{v_i}^{c_1}}$ are public values, and the verifier(s) will never know the value is encrypted from 5.

A.2 The sum of all encrypted values is equivalent to the encrypted value from P

We present the proofs generation and verification for the sum of all encrypted points in B_{v_1}, where we assume $P = 10$, and there are 3 candidates C_1, C_2 and C_3. Voter V_1 cast a vote as:

$$E(5, pk, y_{v_1}^{c_1}, x_{v_1}^{c_1}) = (y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot g^{r_1}, g^5 \cdot pk^{r_1}$$

$$E(2, pk, y_{v_1}^{c_2}, x_{v_1}^{c_2}) = (y_{v_1}^{c_2})^{x_{v_1}^{c_2}} \cdot g^{r_2}, g^2 \cdot pk^{r_2}$$

$$E(3, pk, y_{v_1}^{c_3}, x_{v_1}^{c_3}) = (y_{v_1}^{c_3})^{x_{v_1}^{c_3}} \cdot g^{r_3}, g^3 \cdot pk^{r_3}$$

Prover

- multiply them as

$$\begin{aligned}
&E(5, pk, y_{v_1}^{c_1}, x_{v_1}^{c_1}) \times E(2, pk, y_{v_1}^{c_2}, x_{v_1}^{c_2}) \times E(3, pk, y_{v_1}^{c_3}, x_{v_1}^{c_3}) \\
=&(y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot (y_{v_1}^{c_2})^{x_{v_1}^{c_2}} \cdot (y_{v_1}^{c_3})^{x_{v_1}^{c_3}} \cdot g^{r_1+r_2+r_3}, g^{5+2+3} \cdot pk^{r_1+r_2+r_3} \\
=&(y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot (y_{v_1}^{c_2})^{x_{v_1}^{c_2}} \cdot (y_{v_1}^{c_3})^{x_{v_1}^{c_3}} \cdot g^{r_4}, g^{10} \cdot pk^{r_4}
\end{aligned}$$

- We use c_1 and c_2 to denote $(y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot (y_{v_1}^{c_2})^{x_{v_1}^{c_2}} \cdot (y_{v_1}^{c_3})^{x_{v_1}^{c_3}} \cdot g^{r_4}$ and $g^{10} \cdot pk^{r_4}$, respectively.
- compute $z = \prod_{j=1}^{n_c} (y_{v_1}^{c_j})^{x_{v_1}^{c_j}}$. In this case, $z = (y_{v_1}^{c_1})^{x_{v_1}^{c_1}} \cdot (y_{v_1}^{c_2})^{x_{v_1}^{c_2}} \cdot (y_{v_1}^{c_3})^{x_{v_1}^{c_3}}$
- selects random numbers $k_1, k_2, k_3 \in \mathbb{Z}_q$
- computes $K_1 = (y_{v_1}^{c_1})^{k_1} \cdot (y_{v_1}^{c_2})^{k_2} \cdot (y_{v_1}^{c_3})^{k_3}$
- computes $K_2 = g^{k_1}$, $K_3 = g^{k_2}$
- computes $K_3 = g^{k_2}$
- computes $K_4 = g^{k_3}$
- computes $c = Hash(K_1 \| K_2 \| K_3 \| K_4)$
- computes $Z_1 = x_{v_1}^{c_1} c + k_1$
- computes $Z_2 = x_{v_1}^{c_2} c + k_2$
- computes $Z_3 = x_{v_1}^{c_3} c + k_3$
- PZK(z) = $\{K_1, K_2, K_3, K_4, Z_1, Z_2, Z_3\}$
- Prove $\frac{c_1}{z} (= g^{r_4})$ and $\frac{c_2}{g^{10}} (= pk^{r_4})$ has the same exponentiation (refer to Section 2.1.3)
  - selects $t \in Z$
  - computes $T_1 = (g)^t$
  - computes $T_2 = pk^t$
  - computes $c = Hash(T_1 || T_2)$

  - computes $s = r_4 \cdot c + t$ (r_4 in this case)
  - PZK(P) = $\{T_1, T_2, s, z, \text{ZKP}(z)\}$.
- sends PZK(z) and PZK(P) to **Verifier**

Verifier

- firstly verify z using ZKP(z), compute $c = Hash(K_1 \| K_2 \| K_3)$
  - verify if $(y_{v_1}^{c_1})^{Z_1}(y_{v_1}^{c_2})^{Z_2}(y_{v_1}^{c_3})^{Z_3} = K_1 \times (z)^c$
  - verify if $(y_{v_1}^{c_1})^{Z_1} = K_1 \times (g^{x_{v_1}^{c_1}})^c$
  - verify if $(y_{v_1}^{c_2})^{Z_2} = K_2 \times (g^{x_{v_1}^{c_3}})^c$
  - verify if $(y_{v_1}^{c_3})^{Z_3} = K_3 \times (g^{x_{v_1}^{c_3}})^c$
- secondly verify P using T_1, T_2, s and z
  - multiplies $E(5,\cdots), E(2,\cdots)$ and $E(3,\cdots)$ as (c_1, c_2)
  - computes $c = Hash(T_1 || T_2)$
  - verifies if $g^s = (\frac{c_1}{z})^c \cdot T_1$
  - verifies if $pk^s = (\frac{c_2}{g^{10}})^c \cdot T_2$
  - Same as to verify $\frac{c_1}{z}$ and $\frac{c_2}{g^{10}}$ has the same exponentiation r_4.

Appendix B

Our proposed encryption mechanism for decentralized voting

Here we propose a new encryption mechanism for our election protocols. Note that it is not enough to use the classical ElGamal encryption to protect the privacy of the voters. Indeed, if only ElGamal encryption is used, and if the votes are encrypted by voters public key, then only the voter can reveal the final result. If the content is encrypted by a voter's private key, then everyone can view the content as the user's public key is in the public domain. Finally, if the content of the votes is encrypted by a third party's public key, then, the third party can reveal the content of all users' transactions, which is not suitable for our decentralised protocols.

This is why, to achieve appropriate protection of the voter's privacy, here we propose a new encryption mechanism, which merges ElGamal encryption and group-based encryption into one scheme. The content of votes is encrypted using each voter's private key and is masked by applying public keys of the candidates, as explained in this section.

This new encryption mechanism achieves the following advantages. First, the privacy of all votes is ensured. No one can reveal any vote without the private key of the user and the private key of the corresponding candidate. We assume that all users and all candidates are going to keep

their keys confidential. Second, even the person voting last cannot determine the final outcome before submitting their vote. The final outcome can be computer without revealing contents of votes by using the homomorphic property. We assume that the candidates will apply their private keys after the deadline for the submission of votes, when all votes are submitted and the final outcome is already computed in such a way that everyone can verify the validity of all calculations in the blockchain database.

Denote by $p_{v_i}^{c_j}$ the score assigned by voter V_i to candidate C_j. Applying ElGamal encryption and using the public key pk_{c_j} of candidate C_j, we get the ciphertext

$$E(p_{v_i}^{c_j}, pk_{c_j}) = g^r, g^{(p_{v_i}^{c_j})}(pk_{c_j})^r.$$

After that, we apply group-based encryption (cf. Section 2.1.4) to mask the first part of the ElGamal ciphertext. Namely, the first part g^r of the encrypted value is "encrypted" again by using the pre-computed value y_{v_i} explained in Section 2.1.4 and the private voting key x_{v_i} of the voter V_i, according to the mapping

$$g^r \rightarrow (y_{v_i})^{x_{v_i}} \cdot g^r.$$

This means that each score $p_{v_i}^{c_j}$ is encrypted by applying the following combined formula.

$$E(p_{v_i}^{c_j}, pk_{c_j}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} g^r, g^{(p_{v_i}^{c_j})} \cdot (pk_{c_j})^r. \tag{B.1}$$

where the score $p_{v_i}^{c_j}$ is encrypted by using y_{v_i} (voting public keys of all voters), pk_{c_j} (public key of the score's recipient) and x_{v_i} (the voting private key of the voter).

B.1 An example of proof generation for an encrypted value

Next, we present the proofs generation for an encrypted score in vote B_i, where we assume an encrypted score is $E(1, pk_{c_1}, y_{v_i}, x_{v_i}) = (y_{v_i})^{x_{v_i}} \cdot g^{r_1}, g^1 \cdot (pk_{c_1})^{r_1}$:

Generating proofs for $E(1, pk_{c_1})$:

1. The score is between 1 and 3 (cf. Section 2.1.2):

- generates random numbers $t_1, t_2, t_3, v_2, v_3 \in \mathbb{Z}_q$
- computes $s_2 = t_2 + r \cdot v_2$
- computes $s_3 = t_3 + r \cdot v_3$
- computes $T_{01} = g^{t_1}/(y_{v_i})^{x_{v_i}}$
- computes $T_{02} = g^{t_2}/(y_{v_i})^{x_{v_i}}$
- computes $T_{03} = g^{t_3}/(y_{v_i})^{x_{v_i}}$
- computes $T_1 = {pk_{c_1}}^{t_1}$
- computes $T_2 = (g^{2 \cdot v_2} \cdot (pk_{c_1})^{s_2})/{c_2}^{v_2}$
- computes $T_3 = (g^{3 \cdot v_3} \cdot (pk_{c_1})^{s_3})/{c_2}^{v_3}$
- computes $v = \text{Hash}(c_1 \| c_2 \| T_{01} \| T_{02} \| T_{03} \| T_1 \| T_2 \| T_3)$
- computes $v_1 = v \oplus v_2 \oplus v_3$, where $\oplus$ denotes XOR.
- computes $s_1 = t_1 + r \cdot v_1$
- PPK($p^1_{c_1}$): $\{T_{01}, T_{02}, T_{03}, v_1, v_2, v_3, s_1, s_2, s_3\}$

2. The score $E(1, pk_{c_1}, y_{v_i}, x_{v_i})$ is computed correctly via Equation 6.1 using private key x_{v_i} (cf. Section 2.1.3):

- generates a random number $k_1, k_2 \in \mathbb{Z}_q$
- computes $K_1 = (y_{v_1})^{k_1} \cdot g^{k_2}$
- computes $K_2 = g^{k_1}$
- computes $c = \text{Hash}(K_1 \| K_2)$
- computes $Z_1 = x_{v_i} c + k_1$
- computes $Z_2 = r_1 c + k_2$
- PZK(x_{v_i}): $\{K_1, K_2, Z_1, Z_2\}$

B.2 An example of proof verification for an encrypted value

Next, we give the computation details of verifying PPK($p_{v_i}^{c_j}$) and PZK(x_{v_i}). In the verification of the proofs for $E(1, pk_{c_1})$, we treat $E(1, pk_{c_1}, y_{v_i}, x_{v_i}) = (c_1, c_2)$.

1. Verification of PPK($p_{v_i}^{c_1}$)

   - computes $T_1 = (g^{1 \cdot v_1} \cdot {pk_{c_1}}^{s_1}) / {c_2}^{v_1}$
   - computes $T_2 = (g^{2 \cdot v_2} \cdot {pk_{c_2}}^{s_2}) / {c_2}^{v_2}$
   - computes $T_3 = (g^{3 \cdot v_3} \cdot {pk_{c_3}}^{s_3}) / {c_2}^{v_3}$
   - verify if $v_1 \oplus v_2 \oplus v_3 = \text{Hash}(E(1, pk_{c_1}, y_{v_i}, x_{v_i}) \| \ T_{01} \| T_{02} \| T_{03} \| T_1 \| T_2 \| T_3)$
   - verify if $g^{s_1} = T_{01} \cdot {c_1}^{v_1}$
   - verify if $g^{s_2} = T_{02} \cdot {c_1}^{v_2}$
   - verify if $g^{s_3} = T_{03} \cdot {c_1}^{v_3}$

2. Verification of PZK(x_{v_i})

   - verify if $(y_{v_1})^{Z_1} g^{Z_2} = K_1 \times (c_1)^{\text{Hash}(K_1 \| K_2)}$
   - verify if $g^{Z_1} = K_2 \times (g^{x_{v_i}})^{\text{Hash}(K_1 \| K_2)}$

where y_{v_i} and $g^{x_{v_i}}$ are public values.

Bibliography

[AARTAD17] Anees Ara, Mznah Al-Rodhaan, Yuan Tian, and Abdullah Al-Dhelaan. A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems. *IEEE Access*, 5:12601–12617, 2017.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[Adi12] B. Adida. Helios v4. `http://documentation.heliosvoting.org/verification-specs/helios-v4`, 2012.

[Adi16] J. Adinolfi. And 2016's best-performing commodity is ... bitcoin? `http://www.marketwatch.com/story/and-2016s-best-performing-commodity-is-bitcoin-2016-12-22`, 2016.

[ADMP+09] Ben Adida, Olivier De Marneffe, Olivier Pereira, Jean-Jacques Quisquater, et al. Electing a university president using open-audit voting: Analysis of real-world use of helios. *EVT/WOTE*, 9(10), 2009.

[ALPL13] R Michael Alvarez, Ines Levin, Julia Pomares, and Marcelo Leiras. Voting made safe and easy: the impact of e-voting on citizen perceptions. *Political Science Research and Methods*, 1(1):117–137, 2013.

[AP02] Alan Agresti and Brett Presnell. Misvotes, undervotes and overvotes: The 2000 presidential election in florida. *Statistical Science*, pages 436–440, 2002.

[BBB+13] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Gail Fisher, Philip Kortum, Neal McBurnett, Julian Montoya, Michelle Parker, et al. Star-vote: A secure, transparent, auditable, and reliable voting system. *USENIX Journal of Election Technology and Systems (JETS)*, 1(1):18–37, 2013.

[Bea16] Adrian Beaumont. Us election final results: how trump won. `http://theconversation.com/us-election-final-results-how-trump-won-69356`, December 17, 2016.

[BF05] Steven J Brams and Peter C Fishburn. Going from theory to practice: the mixed success of approval voting. *Social Choice and Welfare*, 25(2-3):457–474, 2005.

[BHRC17] Amit Banerjee, Mahamudul Hasan, Md Auhidur Rahman, and Rajesh Chapagain. Cloak: A stream cipher based encryption protocol for mobile cloud computing. *IEEE Access*, 5:17678–17691, 2017.

[Bit16] BitCongress. BitCongress - Process For Blockchain Voting & Law. `http://www.bitcongress.org/`, 2016.

[Blo] Blockgeeks. Smart Contracts: The Blockchain Technology That Will Replace Lawyers. `https://blockgeeks.com/guides/smart-contracts/`.

[Blo17a] Blockchain.info. Bitcoin block explorer blockchain. `https://blockchain.info/`, 2017.

[Blo17b] Blockchain.info. What if blockchain technology revolutionised voting? scientific foresight unit (stoa), european parliamentary research service, sept. 2016. `https://blockchain.info/charts/n-transactions?timespan=1year`, 2017.

[Bou16] P Boucher. What if blockchain technology revolutionised voting? scientific foresight unit (stoa), european parliamentary research service, sept, 2016.

[BPS12] Michael Brenner, Henning Perl, and Matthew Smith. Practical applications of homomorphic encryption. In *SECRYPT*, pages 5–14, 2012.

[Bre16] Ken Bredemeier. How did trump win election while losing popular vote? https://www.voanews.com/a/how-didi-trump-win-election-while-losing-popular-vote/3591226.html, Nov 11, 2016.

[BSE17] Khadijeh Bagheri, Mohammad-Reza Sadeghi, and Taraneh Eghlidos. An efficient public key encryption scheme based on qc-mdpc lattices. *IEEE Access*, 5:25527–25541, 2017.

[But15] Vitalik Buterin.A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper, 2015.

[But16] Vitalik Buterin. Devcon2: Ethereum in 25 minutes. https://www.youtube.com/watch?v=66SaEDzlmP4, 2016.

[CCC+10] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L Rivest, Emily Shen, et al. Scantegrity ii municipal election at takoma park: the first e2e binding governmental election with ballot privacy. 2010.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 72–83. Springer, 1996.

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

[Cha12] Jonathan Chait. Yes, Bush v. Gore Did Steal the Election. http://nymag.com/daily/intelligencer/2012/06/yes-bush-v-gore-did-steal-the-election.html, 2012.

[Che18] R.-Y. Chen. A traceability chain algorithm for artificial neural networks using T-S fuzzy cognitive maps in blockchain. *Future Generation Computer Systems*, 80:198–210, 2018.

[CLF17] Liquan Chen, Ming Lim, and Zijuan Fan. A public key compression scheme for fully homomorphic encryption based on quadratic parameters with correction. *IEEE Access*, 5:17692–17700, 2017.

[CP92] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Annual International Cryptology Conference*, pages 89–105. Springer, 1992.

[Del17] Deloitte. Blockchain technology: 9 benefits & 7 challenges. `https://blog.deloitte.com.ng/blockchain-technology-benefits-challenges/`, 2017.

[Des16] Jeff Desjardins. It's official: Bitcoin was the top performing currency of 2015. `http://money.visualcapitalist.com/its-official-bitcoin-was-the-top-performing-currency-of-2015/`, 2016.

[DLL12] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *European Symposium on Research in Computer Security*, pages 451–468. Springer, 2012.

[ECH12] Aleksander Essex, Jeremy Clark, and Urs Hengartner. Cobra: Toward concurrent ballot authorization for internet voting. In *EVT/WOTE*, page 3, 2012.

[ECMC16] Christian Esposito, Aniello Castiglione, Ben Martini, and Kim-Kwang Raymond Choo. Cloud manufacturing: security, privacy, and forensic concerns. *IEEE Cloud Computing*, 3(4):16–22, 2016.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[fCR16] International Association for Cryptologic Research. About the helios system. `https://www.iacr.org/elections/eVoting/about-helios.html`, 2016.

[Fol16] Followmyvote.com. Introducing a secure and transparent online voting solution for the modern age: Follow My Vote. `https://followmyvote.com/`, 2016.

[Gar16] DA López García. A flexible e-voting scheme for debate tools. *Computers & Security*, 56:50–62, 2016.

[GKV+16] Dawid Gaweł, Maciej Kosarzecki, Poorvi L Vora, Hua Wu, and Filip Zagórski. Apollo–end-to-end verifiable internet voting with recovery from vote manipulation. In *International Joint Conference on Electronic Voting*, pages 125–143. Springer, 2016.

[GMP] GMPY2.Welcome to gmpy2's documentation! https://gmpy2.readthedocs.io/en/latest/.

[Gre16] M. Gregory. Electronic voting may be faster but carries security risks. http://www.theaustralian.com.au/business/technology/opinion/electronic-voting-may-be/faster-but-carries-security-risks/news-story/f0b6b44844214605e3860ef1887b2bb9, 2016.

[HA13] Hanady Hussien and Hussien Aboelnaga. Design of a secured e-voting system. In *Computer Applications Technology (ICCAT), 2013 International Conference on*, pages 1–5. IEEE, 2013.

[HBB12] Omar Hasan, Lionel Brunie, and Elisa Bertino. Preserving privacy of feedback providers in decentralized reputation systems. *Computers & Security*, 31(7):816–826, 2012.

[Her15] A. Hertig. The first Bitcoin voting machine is on its way. http://motherboard.vice.com/read/the-first-bitcoin-votingmachine-is-on-its-way, 2015.

[Hig16] S. Higgins. Abu Dhabi Stock Exchange Launches Blockchain Voting. https://www.coindesk.com/abu-dhabi-exchange-blockchain-voting/, 2016.

[HRZ10] Feng Hao, Peter YA Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 539–556. Springer, 2000.

[Ket01] Martin Kettle. Recount shows Gore had won. https://www.theguardian.com/world/2001/mar/12/uselections2000.usa, 2001.

[KKJ+16] Yonggon Kim, Ohmin Kwon, Jinsoo Jang, Seongwook Jin, Hyeongboo Baek, Brent Byunghoon Kang, and Hyunsoo Yoon. On-demand bootstrapping mechanism for isolated cryptographic operations on commodity accelerators. *Computers & Security*, 62:33–48, 2016.

[LaC16] Kim LaCapria. Did trump win 3,084 of 3,141 counties, clinton only 57? `https://www.snopes.com/fact-check/trump-won-3084-of-3141-counties-clinton-won-57/`, 2 December 2016.

[LJC+17] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, pages In Press, DOI 10.1016/j.future.2017.08.020, 2017.

[LK16] J. Lavelle and D. Kozaki. Electronic voting has advantages but remains vulnerable to security, software problems. `http://www.abc.net.au/news/2016-07-11/electronic-voting-has-support-but-security-fears-remain/7587366`, 2016.

[LLS+16] Joseph K Liu, Kaitai Liang, Willy Susilo, Jianghua Liu, and Yang Xiang. Two-factor data security protection mechanism for cloud storage system. *IEEE Transactions on Computers*, 65(6):1992–2004, 2016.

[LMW17] Zengpeng Li, Chunguang Ma, and Ding Wang. Towards multi-hop homomorphic identity-based proxy re-encryption via branching program. *IEEE Access*, 5:16214–16228, 2017.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[LVW14] Wenjun Lu, Avinash L Varna, and Min Wu. Confidentiality-preserving image search: a comparative study between homomorphic encryption and distance-preserving randomization. *IEEE Access*, 2:125–141, 2014.

[MMS16] Víctor Mateu, Josep M Miret, and Francesc Sebé. A hybrid approach to vector-based homomorphic tallying remote voting. *International Journal of Information Security*, 15(2):211–221, 2016.

[MMS+17] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304, 2017.

[MNX+16] Abid Mehmood, Iynkaran Natgunanathan, Yong Xiang, Guang Hua, and Song Guo. Protection of big data privacy. *IEEE access*, 4:1821–1834, 2016.

[MOR16] EREN MORENO. Did you know? trump won 3,084 out of 3,151 counties. hillary only won 57. `http://archive.is/L3Dbc`, 1 December 2016.

[MPRJ10] Reshef Meir, Maria Polukarov, Jeffrey S Rosenschein, and Nicholas R Jennings. Convergence to equilibria in plurality voting. In *Proc. of 24th Conference on Artificial Intelligence (AAAI-10)*, pages 823–828, 2010.

[MSH17] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[Mur17] Kevin Murnane. Speedtest Ranks Internet Access Speed In More Than 100 Countries. `https://www.forbes.com/sites/kevinmurnane/2017/08/14/speedtest-ranks-internet-access-speed-in-more-than-100-countries/#335534f065b5`, 2017.

[MV17] Ricardo Mendes and João P Vilela. Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access*, 5:10562–10582, 2017.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *academia.edu*, 2008.

[NCS14] Vu Duc Nguyen, Yang-Wai Chow, and Willy Susilo. On the security of text-based 3d captchas. *Computers & Security*, 45:84–99, 2014.

[Nef04] C Andrew Neff. Practical high certainty intent verification for encrypted votes, 2004.

[NSGF16] Ricardo Neisse, Gary Steri, Dimitris Geneiatakis, and Igor Nai Fovino. A privacy enforcing framework for android applications. *computers & security*, 62:257–277, 2016.

[Par07] Michael Parenti. The Stolen Presidential Elections. `http://www.michaelparenti.org/stolenelections.html`, 2007.

[PP99] Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In *Advances in Cryptology-ASIACRYPT'99*, pages 165–179. Springer, 1999.

[PRB+17] Raj R Parmar, Sudipta Roy, Debnath Bhattacharyya, Samir Kumar Bandyopadhyay, and Tai-Hoon Kim. Large-scale encryption in the hadoop environment: Challenges and solutions. *IEEE Access*, 5:7156–7163, 2017.

[PSO11] Adewole A Philip, Sodiya Adesina Simon, and Arowolo Oluremi. A receipt-free multi-authority e-voting system. *International Journal of Computer Applications*, 30(6):15–23, 2011.

[RBH+09] Peter YA Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *IEEE transactions on information forensics and security*, 4(4):662–673, 2009.

[RFB17] Bahman Rashidi, Carol Fung, and Elisa Bertino. Android resource usage risk assessment using hidden markov model and online learning. *Computers & Security*, 65:90–107, 2017.

[Rob16] Dan Roberts. Why hillary clinton lost the election: the economy, trust and a weak message. `https://www.theguardian.com/us-news/2016/nov/09/hillary-clinton-election-president-loss`, Nov 9, 2016.

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sab16] Samira Saba. Now you can vote online with a selfie. `http://www.businesswire.com/news/home/20161017005354/en/VoteOnline-Selfie`, 2016.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[SCM08] Altair O Santin, Regivaldo G Costa, and Carlos A Maziero. A three-ballot-based secure electronic voting system. *IEEE Security & Privacy*, 6(3):14–21, 2008.

[SP18] P. K. Sharma and J. H. Park. Blockchain based hybrid network architecture for the smart city. *Future Generation Computer Systems*, pages In Press, DOI 10.1016/j.future.2018.04.060, 2018.

[Spe17] Speedtest Global Index. Ranking mobile and fixed broadband speeds from around the world on a monthly basis. http://www.speedtest.net/global-index, December 2017.

[Sza96] Nick Szabo. Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, 1996.

[Top10] Jurij Toplak. Preferential voting: definition and classification. In *Annual Meeting of the Midwest Political Science Assication 67th Annual National Conference*, 2010.

[TPLT13] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From helios to zeus. In *EVT/WOTE*, 2013.

[Wes16] Taylor Wessing. How secure is blockchain? https://www.taylorwessing.com/download/article-how-secure-is-block-chain.html, 2016.

[WIK] WIKIPEDIA. ElGamal encryption. https://wikipedia.org/wiki/ElGamal_encryption.

[Wil16] Rick Wilking. How vulnerable to hacking is the us election cyber infrastructure? https://theconversation.com/how-vulnerable-to-hacking-is-the-us-election-cyber-infrastructure-63241, July 30, 2016.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[Woo17] G. Wood. Ethereum blockchain mechanism (proof of work). https://i.stack.imgur.com/afWDt.jpg, 2017.

[WPD15] CNN Wade Payson-Denney. So, who really won? What the Bush v. Gore studies showed. http://edition.cnn.com/2015/10/31/politics/bush-gore-2000-election-results-studies/, 2015.

[XMY+17] Rui Xu, Kirill Morozov, Yanjiang Yang, Jianying Zhou, and Tsuyoshi Takagi. Efficient outsourcing of secure k-nearest neighbour query over encrypted database. *Computers & Security*, 69:65–83, 2017.

[YO11] Xun Yi and Eiji Okamoto. Practical remote end-to-end voting scheme. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 386–400. Springer, 2011.

[YPB14] Xun Yi, Russell Paulet, and Elisa Bertino. *Homomorphic encryption and applications*, volume 3. Springer, 2014.

[YYR+17] Xuechao Yang, Xun Yi, Caspar Ryan, Ron van Schyndel, Fengling Han, Surya Nepal, and Andy Song. A verifiable ranked choice internet voting system. In *International Conference on Web Information Systems Engineering*, pages 490–501. Springer, 2017.

[ZMH+18] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle. Secure and anonymous decentralized Bitcoin mixing. *Future Generation Computer Systems*, 80:448–466, 2018.

[ZPWZ14] Yingming Zhao, Yue Pan, Sanchao Wang, and Junxing Zhang. An anonymous voting system based on homomorphic encryption. In *Communications (COMM), 2014 10th International Conference on*, pages 1–4. IEEE, 2014.