TRANSPARENT AND MUTUAL RESTRAINING ELECTRONIC VOTING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Huian Li

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Xukai Zou, Co-Chair
    Department of Computer Science
Dr. Ninghui Li, Co-Chair
    Department of Computer Science
Dr. Feng Li
    Department of Computer and Information Technology
Dr. Elisa Bertino
    Department of Computer Science

Approved by:
    Dr. Voicu Popescu by Dr. William J. Gorman
        Head of the Department Graduate Program

To my family for their wholehearted support and constant encouragement.

ACKNOWLEDGMENTS

I owe a great deal of debts to people who have given me the generous help during this long journey.

First and foremost, my co-advisor, Prof. Xukai Zou, opened the door of cryptography for me. Since then, he guided me through all kinds of difficulties with great patience and consideration. He taught me not only the discipline required to conduct research work, but also the knowledge to make life better.

My co-advisor, Prof. Ninghui Li, had deep discussions with me on cryptography and provided several key insights on this field, which literally broaden my horizon. He gave a list of key papers, and also instructed me how to write paper.

Being on my advisory committee, Prof. Feng Li, gave tremendous help on my research work. He routinely joined group discussions and taught how to read papers. He always contributed important ideas, which shaped this work in a significant way.

In addition, Dr. Yan Sui and Dr. Wei Peng had numerous constructive conversations with me about this topic. My work is built upon many of their important suggestions and comments. In particular, Dr. Yan Sui gave me tremendous help on data simulation.

I am grateful to Dr. Elisa Bertino for taking time to serve on my prelim and final exam committees, and her helpful feedback.

I would thank other faculty and staff members at the computer science department and the graduate school for their support and help.

Of course, I like to express my sincere gratitude to many anonymous reviewers who read our papers carefully and gave detailed and constructive ideas. I also thank people that I may have forgotten their names, but helped me to pull off this work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Li, Huian PhD, Purdue University, December 2018. Transparent and Mutual Restraining Electronic Voting. Major Professors: Xukai Zou and Ninghui Li.

Many e-voting techniques have been proposed but not widely used in reality. One of the problems associated with most of existing e-voting techniques is the lack of transparency, leading to a failure to deliver voter assurance. In this work, we propose a transparent, auditable, end-to-end verifiable, and mutual restraining e-voting protocol that exploits the existing multi-party political dynamics such as in the US. The new e-voting protocol consists of three original technical contributions – universal verifiable voting vector, forward and backward mutual lock voting, and in-process check and enforcement – that, along with a public real time bulletin board, resolves the apparent conflicts in voting such as anonymity vs. accountability and privacy vs. verifiability. Especially, the trust is split equally among tallying authorities who have conflicting interests and will technically restrain each other. The voting and tallying processes are transparent to voters and any third party, which allow any voter to verify that his vote is indeed counted and also allow any third party to audit the tally. For the environment requiring receipt-freeness and coercion-resistance, we introduce additional approaches to counter vote-selling and voter-coercion issues. Our interactive voting protocol is suitable for small number of voters like boardroom voting where interaction between voters is encouraged and self-tallying is necessary; while our non-interactive protocol is for the scenario of large number of voters where interaction is prohibitively expensive. Equipped with a hierarchical voting structure, our protocols can enable open and fair elections at any scale.

1. INTRODUCTION

1.1 Problem Statement

Voting is the pillar of modern democracies. Traditional voting, however, suffers from both low efficiency and unintentional errors. The event surrounding the 2000 US presidential election witnessed the shortcomings of punch-cards and other antiquated voting systems. The Help America Vote Act [1] and the creation of the Election Assistance Commission (EAC) [2] highlighted the determination of the US to deploy more modern voting systems. A survey sponsored by EAC shows that 17.5% of votes in the 2008 US presidential election were cast as absentee ballots [3]. This demonstrates a demand for less centralized voting procedures/booths. One potential solution is to allow voters to cast ballots on Internet-enabled mobile devices [4].

Online voting[1] has been an active research topic with many advantages over traditional voting, but presents some unique challenges. For example, if a discrepancy is found in the tally, votes need to be recounted and the source of the discrepancy needs to be identified. The recounting and investigating should nevertheless preserve votes' anonymity and voters' privacy. Other voting requirements, such as verifiability and receipt-freeness, make the problem even more challenging due to their inherently contradicting nature [5, 6].

Several online voting solutions [7–11] have been proposed. Some suggest keeping non-electronic parallels of electronic votes, or saving copies of votes in portable storage devices. They either fail to identify sources of discrepancy or are susceptible to vote-selling and voter-coercion. Most solutions [6, 7, 12–16] are based on cryp-

---

[1] Online voting, Internet voting, remote voting, and electronic voting (e-voting) are used interchangeably in this dissertation. However, there are some subtle differences in certain literature. For example, e-voting sometimes includes voting with electronic devices in a voting booth; while remote voting refers to voting from anywhere through Internet, without the assumption of a booth.

tographic techniques, such as secret sharing, mix-net, and blind signature. These solutions are often *opaque*: Except casting their votes, voters do not directly participate in collecting and tallying votes, and the voting results are typically acquired through decryption by third parties only, such as talliers. This raises concerns over the trustworthiness and transparency of the entire voting process. In addition, these solutions sometimes entrust the fairness of the voting process onto the impartiality of authorities. Voting under multiple conflict-of-interests parties is not addressed by these solutions.

Furthermore, examination of current voting systems including online voting techniques shows a gap between casting secret ballots and tallying/verifying individual votes. This gap is caused either by the disconnection between the vote-casting process and the vote-tallying process, or by the opaque transition (e.g., due to encryption) from vote-casting to vote-tallying, thus damaging voter assurance with an open question: “Will my vote count?” [17].

1.2 Major Contributions

In this work, we propose a transparent, auditable, and end-to-end verifiable voting protocol that exploits conflict of interests in multiple tallying authorities, such as the two-party political system in the US. It consists of a few novel techniques – universal verifiable voting vector, forward and backward mutual lock voting, and proven in-process check and enforcement – that, in combination, resolves the apparent conflicts such as anonymity vs. accountability and privacy vs. verifiability.

Our main contributions are as follows:

1. Light-weight ballot generation and tallying. The new e-voting protocol needs only (modular) addition and subtraction in ballot generation and vote tallying, rather than encryption/decryption or modular exponentiations. Thus, the newly proposed protocol is efficient.

2. Seamless transition from ballots to plain votes. In our protocol, individual ballots can be aggregated one by one and the final aggregation reveals all individual votes (in their plain/clear format). The aggregation is simply modular addition and can be performed by anyone (with modular addition knowledge and skills) without involvement of a third-party entity. The aggregation has the all-or-nothing effect in the sense that all partial sums reveal no information about individual votes but the final (total) sum exposes all individual votes. Thus, the newly proposed protocol delivers fairness in the voting process.

3. Viewable tallying and verification. The cast ballots, sum(s) of ballots, individual votes, and sum(s) of votes for each candidate are all displayed on a public bulletin board. A voter or anyone can view them, verify them visually, and even conduct summations of ballots (and as well as votes) himself. Thus, the newly proposed protocol delivers individual and universal verifications.

4. Transparency of the entire voting process. Voters can view and verify their ballots, plain votes, and transition from ballots to votes and even perform self-tallying [18, 19]. There is no gap or black-box [20] which is related to homomorphic encryption or mix-net in vote casting, tallying and verification processes. Thus, in the newly proposed protocol, the voting process is straightforward and delivers transparency.

5. Voter assurance. The most groundbreaking feature of our voting protocol, different from all existing ones, is the separation and guarantee of two distinct voter assurances: 1) *vote-casting assurance* on *secret ballots* – any voter is assured that the vote-casting is indeed completed (i.e., the secret ballot is confirmatively cast and viewably aggregated), thanks to the openness of secret ballots and incremental aggregation, and 2) *vote-tallying assurance* – any voter is assured that their vote is visibly counted in the final tally, thanks to the seamless transition from secret ballots having no information to public votes having complete (but anonymous) information offered by the simplified (n, n)-secret sharing scheme. In addition, end-to-end individual verification and universal verification allow the public to verify the accuracy of the count, and political parties to catch fraudulent votes. Thus, the newly proposed protocol delivers full voter assurance.

Real time check and verification of any cast secret ballots and their incremental aggregation, along with transparency, provide strong vote integrity and auditability. Any attacks, regardless of from inside such as collusion among voters and/or a collector or from outside such as external intrusion, which lead to any invalid votes, can be detected via the tallied voting vector. All these properties are achieved while still retaining what is perhaps the core value of democratic elections – the secrecy of any voter's vote.

In particular, we relax the trust assumption a bit in our protocol. Instead of assuming that tallying authorities who conduct tally are trustworthy,[2] the new protocol, like the split trust in the split paper ballot voting [21], splits the trust equally among tallying authorities. As long as one tallying authority behaves honestly, misbehaviors from one or all other tallying authorities will be detected. In addition, as we will analyze later, our protocol is robust against collusion between voters and one tallying authority, and any misbehavior or attack leading to an invalid tallied voting vector will be detected. Thus, in order not to impress readers that we need trusted central authorities, we chose to use a relatively neutral term called *collectors* in the rest of the dissertation. We assume collectors will not collude since they represent parties with conflicting interests.

Receipt-freeness and coercion-resistance are two of most challenging issues in online voting. We address vote-selling by replacing visual verification with individual verification, and adopt the concept of fake credentials to conquer voter-coercion.

We understand the scalability concern with very large number of voters, but this concern can be addressed by incorporating a hierarchical voting structure. In reality, most voting systems [22] present a hierarchical structure such as a tree where voting

---

[2] Some protocols realize trustworthiness (tally integrity) by using commitments and zero-knowledge proof to prevent a tallying authority from mis-tallying and some use threshold cryptography to guarantee the trustworthiness of tallying results (and vote secrecy) as long as t out of n tallying authorities behave honestly.

is first carried out in each precinct (for example, towns or counties) with relatively small number of voters and then vote totals from each precinct are transferred to an upper level central location for consolidation. We will have a more detailed discussion about this later.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 gives an overview of typical components in a voting system, and voting requirements. Chapter 3 introduces a naive voting protocol. Chapter 4 is our mutual restraining voting protocol, together with its complexity analysis and simulation results. Chapter 5 explores the location anonymization schemes used in our protocols. Chapter 6 presents security and property analysis for the protocol in Chapters 4 and 5. Chapter 7 addresses vote-selling and voter-coercion issues that exist in the previous protocols. Chapter 8 shows a prototype implementation of the protocol in Chapter 4. Chapter 9 discusses scalability of our voting protocol. Chapter 10 provides the related work and protocol comparison. Chapter 11 concludes the dissertation and lays out the future work.

2. E-VOTING REQUIREMENTS AND ITS TYPICAL COMPOSITION

In this chapter, we first go through the requirements of an e-voting system, and then provide an overview of typical entities, stages, and cryptographic techniques in e-voting [23].

2.1 Major E-voting Requirements

In order for an e-voting scheme to be useful in practice, it should satisfy several requirements [15, 24–27]. We review the basic requirements first, and then give a list of requirements for countering attacks.

2.1.1 Basic Requirements

We argue that the requirements listed below are fundamental for every voting scheme.

Correctness. If all the participants are honest and following the protocol, the voting results reflect all voters' will.

Privacy. Privacy implies that voter's information should remain secret. This is actually one of voting regulations in certain countries. If a voter's information is published along with his vote, this certainly violates privacy.

Verifiability. A voter can verify his vote without revealing the identity and makes sure that his vote is counted (which is called individual verifiability). Anyone can verify the final voting results (which is called universal verifiability).

Eligibility. Only eligible voters are allowed to vote. This is usually implemented through authentication.

Reliability. The system should work without compromising votes, even if certain system failure occurs.

Transparency. Voters should be able to understand the system generally. If a voting scheme allows a voter to participate in every stages from initial setup to final tally and verification, voters will be able to witness the whole process. Certainly this provides transparency, thus achieving voter assurance. In other words, voters will be assured that his vote is correctly cast and counted through transparency.

Fairness. Information about votes cannot be learned until the voting results are published. Any participants (even outsiders) cannot gain knowledge of the voting results before its final publication, thus precluding preannouncement of any partial voting results.

2.1.2 Counter Attack Requirements

It is important for a voting scheme to have certain counter attack requirements. Such features enhance the security of the voting process. A vulnerable voting scheme may be attacked by adversaries who will try to manipulate the election and lead to incorrect results. A secure voting scheme should possess the following requirements.

Robustness. Robustness of a scheme guarantees resistance against active and passive attacks, and any errors in the voting process. There might be several entities trying to disrupt the voting process, so a robust scheme must provide the required security and allow voters to complete voting without any interruption.

Incoercibility (Coercion-resistance). In order to manipulate the voting results, an adversary may use multiple methods to coerce voters. For example, he can demand a voter to refrain from voting; or he can represent as a valid voter if he gets this voter's private key. So an incoercible voting scheme should defend against such adversary to coerce voters.

Receipt-freeness. A voter should not be provided with a receipt which may be used to trace the vote by aother entity. In particular, a receipt may be used as a

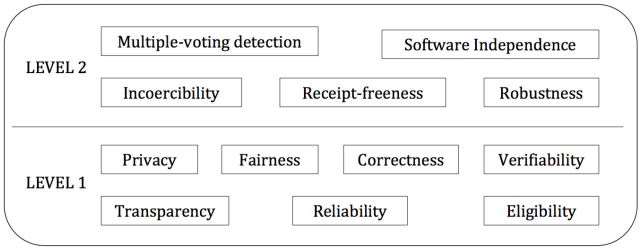


Fig. 2.1. Requirements of e-voting scheme

proof to show the vote is cast as requested for vote buying and selling, and also for voter-coercion.

Multiple-voting Detection. Multiple voting by a single voter should be identified. In a voting system allowing multiple votes, there are two approaches in literature [6,13] to address this issue. One approach is that the last vote counts. In another approach, if all multiple votes by one voter devote to the same candidate, only one is counted. However, if these votes go to different candidates, all votes by this voter are cancelled. These counting approaches are usually used to defend voter-coercion.

Software Independence. Software independence requires that an undetected change or error cannot cause an undetectable change or error in an election outcome [28]. This usually requires thorough evaluation and examination of the voting scheme.

Fig. 2.1 shows two levels of requirements of a voting scheme in which the first level is fundamental, while the second level includes counter attack and advanced requirements, and the requirements in higher level are more difficult to implement. In general, all the requirements are equally important for remote elections. However, depending on the voting situation and certain specific needs, some requirements may outweigh others. As pointed out in [6], there is no single voting scheme satisfying all requirements.

From a different perspective, several design principles for secure remote voting are listed in [29] including proven security, trustworthy design responsibility, published source code, vote verification, voter accessibility, ensuring anonymization, and expert oversight. Most of them correspond to the requirements mentioned above.

2.2 Typical Remote Voting Entities

Here we enumerate all entities that will be involved in an e-voting process. A typical e-voting scheme includes the following entities.

Voter. Voter is a person who will choose among the contesting candidates and cast vote. By using an authentication system, all schemes should be able to tell an eligible voter from ineligible ones. In schemes that considering voter-coercion, a coercer could pretend to be a voter.

Candidate. Candidate is a person contesting in the election. There are different positions to which a candidate may contest. The list of all the contestants or candidates will be shown to a voter and the voter has to decide for which candidate he has to vote.

Authority. Authority is an entity responsible for conducting the elections. The authority has to follow all the guidelines and implement the protocols for voting. Typically, a voting system will have multiple authorities.

Registrar. Registrar in a voting scheme is responsible for authenticating voters. Usually a set of registrars is assumed. They jointly issue keying materials such as private keys and public keys to voters.

Auditor. Auditor will audit the voting process by inspecting individual votes, final voting results, or voting logs.

Adversary. Apart from the entities above, there can be a malicious entity in the voting model called adversary which will attempt to manipulate the voting process and results. There are two types of adversary, *external* and *internal*. The external adversary will actively try to coerce a voter or breach the privacy of voters and an internal adversary, apart from breaching the privacy, may also try to corrupt the authority from inside. Adversary can also be classified into *passive* and *active*. The passive adversary honestly follows the protocol but tries to infer more information, while the active adversary aims to violate the protocol in various ways.

Bulletin Board. This is the place publicly accessible to all entities listed above, usually with the appendive-write capability. It can be a public web site even allowing outsiders to access (but not write functionality). In certain schemes, it is a piece of universally accessible memory [6, 13].

However, not all schemes assume the existence of all entities mentioned above. For example, auditor is omitted from many schemes. In some schemes, authority may also carry the duty of registrar.

2.3 Generic Remote Voting Stages

In general, a voting scheme has several stages [30, 31] according to the voting model, and entities participate in different stages to follow cryptographic protocols and satisfy the requirements. The generic stages are given below.

Initialization. The authority or voting center prepares for the remote voting by following an initialization protocol. For example, in the RSA based scheme [32], the authority generates two large prime numbers as public and private keys for the rest of the voting process.

Registration/Authentication. Every voter has to register and be authenticated before casting vote. For example, the identity of a voter is checked using a browser based cryptographic protocol or by an email.

Vote Casting. The voter casts his vote using the exclusive credentials or keys given by the authority.

Vote Tallying. The authorities tally all the votes and publish the voting results. Typically votes are anonymous and sometimes encrypted.

Vote Verification. The voter should be provided with a verification procedure such that he can verify which candidate he has voted for.

Auditing. Auditing is done by the authority after finishing the voting procedure. The vote count is determined and will be sent to a central authority.

The above are generic stages followed by most of remote voting schemes. There might be schemes that have some additional or less stages.

2.4 Assumptions and Building Blocks

In this section, we first give some common cryptographic assumptions. A few common building blocks are presented afterwards.

2.4.1 Common Assumptions

Untappable Channel. Several remote voting schemes, especially early ones, assume an untappable channel between communication parties, typically, voters and authorities (or registrars). It provides information-theoretic secrecy, but is not practical in reality. Certain scheme even makes stronger and unrealistic assumption such as an anonymous untappable channel.

Anonymous Channel. Anonymous channel is relaxed from untappable channel in terms of security, by allowing adversaries to spy on the communication channel and to intercept data. This is more realistic assumption about the distribution of credentials by registrars or authorities.

Voting Booth. Many remote voting schemes assume the existence of voting booth. The voting booths are governed by authorities, some even with guard. Usually only one voter is allowed to enter booth at a time and communication channels are provided between voting authorities and a booth so a voter can cast his vote. Typically no receipt is printed after a vote is casted. (In case a voter does receive a receipt, he will be asked to destroy it before leaving the booth.) This assumption provides receipt-freeness. However, having physical booths is not a practical assumption for remote voting.

2.4.2 Common Building Blocks

Deniable Encryption. Deniable encryption is used against revealing encrypted information so that the owner of this information can decrypt it in an alternative way to different plaintext. It was introduced in [33, 34] that allows a sender to encrypt a bit b in such a way that the resulting ciphertext can be explained as either b or $1-b$ to a coercer.

Depending on which party being coerced, deniable encryption is classified into sender-deniable scheme (resilient against coercing, i.e., demanding to see the sender's ciphertext), receiver-deniable scheme, and sender-and-receiver-deniable scheme. Based on keys used between a sender and a receiver, it is also classified into public key deniable encryption and shared key deniable encryption. A sender-deniable public key encryption based on RSA was proposed in [15], and a receiver-deniable encryption scheme based on ElGamal was introduced in [35].

Zero-knowledge Proof. Zero-knowledge proof (ZKP) is also frequently used in various stages of a e-voting scheme between senders and receivers for verification purposes. It is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any additional information apart from the fact that the statement is indeed true. For cases where the ability to prove the statement requires some secret information on the part of the prover, the definition implies that the verifier will not be able to prove the statement to anyone else.

It has two forms, non-interactive ZKP and interactive ZKP. The first form is used primarily in voting schemes.

Secure Multi-party Computation. Secure multi-party computation is adopted into e-voting schemes to allow participants to carry computation jointly without anyone disclosing his data to others. Yao [36] has shown that any multi-party computation can be performed using a garbled circuit representation, but it is hard to implement efficiently. In this dissertation, we utilize secure two-party multiplication proposed by Samet and Miri [37].

Plaintext Equivalence Test. Plaintext equivalence test [38] is usually used to check if two encrypted votes are identical. Given two ciphertexts $\{v_1\}_k^{r_1}$ and $\{v_2\}_k^{r_2}$ respectively, with each encrypted using the same key k, a plaintext equivalence test allows the holders of the decryption key to demonstrate that plaintexts v_1 and v_2 are equal without revealing the decryption key or any information about v_1 and v_2.

Certainly, there are many other building blocks, such as randomizer, commitment schemes, and Blockchain.

2.5 Primary Cryptographic Techniques

Here we present four primary cryptographic techniques including mix-net, blind signature, threshold homomorphic encryption, and secret sharing.

2.5.1 Mix-net

Mix-net is introduced by David Chaum [39] as a technique to provide anonymous communication, and later is applied into many e-voting schemes [13, 40, 41]. It is based on public key cryptography to provide anonymity and untraceability. Mix-net is multiparty communication protocol which takes input messages, shuffles them in a random order such that all the parties know that shuffling is performed but no party knows about shuffling algorithm.

In today's network, a sender and a receiver no longer remain confidential because every packet transmitted consists of IP addresses of the sender and the receiver. Anyone can look through the packets to gain knowledge of the sender and the receiver. By using anonymous channels we can hide the information of sender so that even the receiver cannot relate back to the sender. In mix-net, such an anonymous communication is implemented by a set of nodes which take messages as inputs and bounce them back in a shuffled order [42].

When a sender wants to send a message, he passes it on to a node. Then the node permutes and passes the message to the next node, the last node sends the message to the receiver. As long as one node is honest and functions correctly, the anonymity of the sender can be guaranteed. Mix-net can be implemented in two categories:

2.5.1.1 Decryption Mix-net

In this type of mix-net, the nodes have a pair of public and private keys. A public key infrastructure is used to distribute the keys [40,42]. Let pub_i be the public key and $priv_i$ the private key for the i-th node, and r_i be a random padding. The encryption protocol works as follows if a voter sends a message v through five nodes:

$$v_{enc} = E_{pub_1}(r_1, E_{pub_2}(r_2, E_{pub_3}(r_3, E_{pub_4}(r_4, E_{pub_5}(r_5, v)))))$$

Here the message will be encrypted in layers, the encrypted messages will pass through the nodes in the correct order, the nodes will decrypt the message and the last node delivers the message v. The decryption protocol works similarly by using the private keys.

2.5.1.2 Re-encryption Mix-net

Re-encryption mix-net also consists of multiple nodes to randomize and pass the messages. In this process, instead of decrypting the message from previous node, each node re-encrypts the message and passes on to the next node. Therefore if one node is honest we can guarantee that message is anonymous. The re-encryption mix-net can be deployed using different cryptosystems. One example is the ElGamal cryptosystem [40, 42].

2.5.2 Blind Signature

Blind signature [43] is fundamentally one kind of digital signature where the content of a message is blinded before it is signed. In another word, it allows a person to get a third party to sign a message without revealing the content of a message, thus achieving confidentiality of the voter's ballot. Usually an authority blindly signs a voter's vote to authenticate it. Hence the authority whose function is to verify the eligibility of a voter will not know whom the voter votes for.

Currently blind key signature schemes are present with many public key protocols. One of such schemes is blind RSA scheme in which a traditional RSA signature is used. Here is a simple scheme of blind signature based on RSA signing [44]. Let (N, e) be the authority's public key and (N, d) be his private key where d is the inverse of $e \bmod \phi(N)$. A voter chooses a random number r such that $gcd(r, N) = 1$, and sends the following to the authority:

$$v' = v \cdot r^e \bmod N$$

The random number r is used to hide the ballot v from the authority. The authority then signs the blinded ballot after verification and sends back $S^{'}$.

$$S' = (v')^d = v^d \cdot (r^e)^d = v^d \cdot r \bmod N$$

After receiving $S^{'}$, the voter unblinds it to get the true signature S since he knows r.

$$S = S' \cdot r^{-1} = v^d \cdot r \cdot r^{-1} = v^d \bmod N$$

Anonymous channels can be used to provide maximum privacy. A voter will get a blind signed vote, and then submit vote to the mix-net. After all votes are cast, the mix-net will process the encrypted votes. The authority decrypts the votes shuffled by mix-net and displays the result to public. This approach is efficient but lacks transparency.

2.5.3 Threshold Homomorphic Encryption

Homomorphic encryption was first proposed by Benaloh et al. [45, 46]. It refers to a scenario in which the encryption of combined secret can be reconstructed from multiple independently encrypted secrets. Let the operations $\oplus$ and $\otimes$ be defined on the plaintext space and the ciphertext space, respectively. The "product" of the encryptions of two votes v_1 and v_2 is the encryption of the "sum" of two votes v_1 and v_2. More specifically, $E(v_1 \oplus v_2) = E(v_1) \otimes E(v_2)$. Examples of partially homomorphic cryptosystems [47] include ElGamal, Paillier, RSA, and a few others. Below we give further detail of ElGamal and Paillier cryptosystems.

2.5.3.1 ElGamal

The ElGamal cryptosystem [48] is adopted often in e-voting schemes. It is by nature homomorphic with multiplication. Assume in a commutative group G of order $|G| = q$, the public key is (G, q, g, h) where g is a generator of G, $h = g^x$, and x is the secret key. The encryption of a vote v is $E(v) = (\alpha, \beta) = (g^r, v \cdot h^r)$ for some random $r \in \{0, 1, ..., q-1\}$. For two votes which are encrypted as

$$\begin{aligned} E(v_1) &= (\alpha_1, \beta_1) = (g^{r_1}, v_1 \cdot h^{r_1}), \text{ and} \\ E(v_2) &= (\alpha_2, \beta_2) = (g^{r_2}, v_2 \cdot h^{r_2}) \end{aligned}$$

The homomorphic property is then:

$$\begin{aligned} E(v_1) \cdot E(v_2) &= (\alpha_1, \beta_1) \cdot (\alpha_2, \beta_2) \\ &= (g^{r_1}, v_1 \cdot h^{r_1}) \cdot (g^{r_2}, v_2 \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, (v_1 \cdot v_2) h^{r_1+r_2}) = E(v_1 \cdot v_2) \end{aligned}$$

The encrypted votes are "summed" by using the homomorphic property of the encryption function (without decrypting them). However, ElGamal is only multiplicative homomorphic as shown above, so e-voting systems taking advantage of ElGamal usually adopt a slightly modified cryptosystem which is additive homomorphic.

The modified ElGamal works as below. The encryption of a vote v is $E(v) = (\alpha, \beta) = (g^r, p^v \cdot h^r)$ where p is another independent (from g) generator of G. For two votes which are encrypted as

$$\begin{aligned} E(v_1) &= (\alpha_1, \beta_1) = (g^{r_1}, p^{v_1} \cdot h^{r_1}), \text{ and} \\ E(v_2) &= (\alpha_2, \beta_2) = (g^{r_2}, p^{v_2} \cdot h^{r_2}) \end{aligned}$$

So the additive homomorphic property is:

$$\begin{aligned} E(v_1) \cdot E(v_2) &= (\alpha_1, \beta_1) \cdot (\alpha_2, \beta_2) \\ &= (g^{r_1}, p^{v_1} \cdot h^{r_1}) \cdot (g^{r_2}, p^{v_2} \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, p^{v_1+v_2} h^{r_1+r_2}) = E(v_1 + v_2) \end{aligned}$$

2.5.3.2 Paillier

The Paillier cryptosystem [49] is another probabilistic encryption function for public key cryptography. The notable feature is its homomorphic property. In this cryptosystem, assume the public key is the modulus m and the base is g, the encryption of a vote v is $E(x) = g^x r^m \bmod m^2$ for some random $r \in \{0, 1, ..., m-1\}$. For two votes which are encrypted as

$$\begin{aligned} E(v_1) &= g^{v_1} r_1^m, \text{ and} \\ E(v_2) &= g^{v_2} r_2^m \end{aligned}$$

The homomorphic property is then:

$$\begin{aligned} E(v_1) \cdot E(v_2) &= (g^{v_1} r_1^m)(g^{v_2} r_2^m) \\ &= g^{v_1+v_2} (r_1 r_2)^m = E(v1 + v2 \bmod m) \end{aligned}$$

In many e-voting schemes, all votes can be added by using homomorphic encryption without decrypting them first, therefore ensuring privacy. So homomorphic encryption is an efficient method for e-voting schemes.

2.5.4 Secret Sharing

Secret sharing is also used in many e-voting schemes due to its efficiency and simplicity. In Shamir's scheme [50], a secret s in a finite field is partitioned into n shares where any k-sized subset of these n shares reveals s ($k \leq n$), but any subset of size smaller than k reveals nothing about s.

A secret sharing scheme can have the property of homomorphism. As an example, simplified (n, n) secret sharing [51] used in this dissertation is additively homomorphic. Several e-voting schemes [52–54] exploit homomorphism based on secret sharing.

Some schemes [53] utilize Shamir's threshold secret sharing, while some [54] are based on Chinese Remainder Theorem (CRT).

Table 2.1 summarizes the four major cryptographic techniques used in e-voting systems. Currently, most schemes utilize mix-net and homomorphic encryption.

Table 2.1.
Typical cryptographic primitives in e-voting and their comparison

Typical Crypto-primitives	**Advantages**	**Disadvantages**
Mix-net	Shuffling makes votes unlinkable to voters; No fixed sequence of stages is required [55, 56]	Multiple encryptions are needed for input; Large size messages are not efficiently accommodated [55, 56]
Blind signature	Efficient and simple to implement	Signer has no control over attributes except for those bound by public key; Universal verifiability is hard to implement
Homomorphic encryption	Tallying procedure is simple; Votes cannot be tallied before being cast	Susceptible to some attacks such as RSA blinding attack; Concern over zero knowledge proof used in voting schemes
Secret sharing	Increased reliability and confidentiality; Low security demand of communication channel	Robust but harder to implement; Concern over scalability

3. NAÏVE SECRET SHARING BASED VOTING

In this chapter, we provide naïve voting protocol that is based on secret sharing [57]. Below we first present the interactive protocol in detail, and then discuss the non-interactive protocol in Section 3.7.

3.1 Assumptions and Threat Model

First, we describe the assumptions and security models here.

3.1.1 Assumptions

We assume that there are $N + 3$ parties: N ($N > 3$) voters, two conflict-of-interests and honest collectors[1] C_1 and C_2, and a trusted third authority M_{auth}. The collectors collect certain data during the voting process, and publish all voters' individual votes. They also work together with the third authority to find the culprit if a voter misbehaves in the scheme. The third authority steps in only when the vote verification fails, and he performs investigation only.

We assume the network supports both unicast and broadcast. The unicast channel operates securely. Data integrity is enforced. A voter publishes his data through broadcasting.

---

[1] This assumption is reasonable in real life since typically collectors from different political parties will join together to tally all votes, which implicitly enforces mutual monitoring. More collectors in this scheme are also allowed.

3.1.2 Threat Model

Active and *passive* adversaries will threat the operations of protocol. The passive attack happens when the adversary colludes with other voters to infer certain messages based on their own information. On the other hand, the active attack aims to modify messages and eventually to bring down the protocol. Particularly, certain misbehaving voter may send incorrect data to invalidate others' votes. In the protocol here, we will address both types of adversary.

3.2 Building Blocks

Here we describe building blocks used in the voting protocol.

3.2.1 Hash Functions

We use two types of hash functions. The first type is regular hash function, which simply maps one data set in certain range to another set in the same range. For example, if we define a datum of integer with four bytes, the range will be $[-2^{31}, 2^{31}]$. Comparing to the cryptographic hash functions mentioned later, this type is more efficient in computation and easier in implementation.

The second is cryptographic hash function, which typically constructs fingerprint of some data and provides assurance of data integrity [58]. It has properties of preimage resistance, second-preimage resistance, and collision resistance.

3.2.2 Simplified (n, n)-Secret Sharing

Secret sharing (SS) is applied into our e-voting scheme. Polynomial-based threshold SS scheme was first proposed by Shamir [59]. Here we adopt a simplified (n, n)-SS scheme [51]. A secret s is split into n shares s_i $(1 \leq i \leq n)$, $s = \sum_{i=1}^{n} s_i$, over group $\mathbb{Z}_m$ where $m > 2^n$. For a group of n members, each receives one share. All n members need to pool their shares together to recover s [60]. The scheme is additively homomorphic [61]: The sum of two shares $s_i + s'_i$ (corresponding to s and s', respectively) is a share of the secret $s + s'$.

Theorem 3.2.1 *The simplified* (n, n)*-SS scheme is unconditionally indistinguishable. That is, collusion of even up to* $n - 1$ *participants cannot gain any bit of information on the shares of the rest.* The proof is given in [61].

Corollary 3.2.2 *In the simplified* (n, n)*-SS scheme, if* k *shares are known, collusion of even up to* $n-1-k$ *participants cannot gain any bit of information on the unknown shares.*

Proof When k shares are unknown, the simplified (n, n)-SS is degraded to simplified $(n-k, n-k)$-SS. Thus, based on the theorem above, collusion of even up to $n-1-k$ participants cannot find any bit of information on the unknown shares. $\blacksquare$

This applies to the case when some shares are make public.

Corollary 3.2.3 *In the simplified* (n, n)*-SS scheme, if the secret is known, collusion of even up to* $n - 2$ *participants cannot gain any bit of information on the unknown shares. In addition to that, if* k *shares are known, collusion of up to* $n - 2 - k$ *participants cannot gain any information of the remaining shares as long as there are two or more unknown/secret shares.*

Proof Based on the theorem above, because of the randomness of the shares, as long as there are two or more shares are unknown, collusion of even up to $n - 2 - k$ participants cannot find any bit of information on these unknown shares. $\blacksquare$

This corollary applied to the case in our voting protocol that after a vote or ballot is open, collusion of up to $n-2$ voters cannot find any information of unknown shares. If k shares are also known or made public, as long as two or more shares remain secret, collusion of up to $n - 2 - k$ voters cannot find any information of the remaining unknown shares.

3.2.3 Commitment Scheme

We use commitment scheme to allow a member who receives a secret share to verify if his share is consistent with the one in the original secret holder. Commitment scheme has been proposed in several VSS schemes such as [62, 63]. Our scheme is based on cryptographic hash function described in Section 3.2.1. A voter commits himself to x by computing

$$y = H(x)$$

and publishes y as a commitment to x. Any voter receiving x can verify it by computing $H(x)$ and check if $H(x)$ he calculated equals to y that the committer published.

3.2.4 Location Anonymization Scheme

Location anonymization scheme (LAS) is to find a unique position for each voter V_i. The index i of V_i is known to the public. The approach taken here is based on Zhao et al.'s scheme [51]. Given N voters, each voter V_i preloads a location vector P_i first, and then chooses a position j randomly in a vector P_i. All voters aggregate their P_i's using (N, N)-SS. By checking the aggregated result, each one can observe if he obtains a position without colliding with others. If not, another round is needed. The voters who obtain unique positions in this round can keep their previous selections, while others randomly make selections from remaining unoccupied positions. Each voter at the end obtains a unique position. Due to (N, N)-SS, the obtained position by each voter is only known to himself. More details can be found in [51].

In addition, we can also use the schemes developed in Chapter 5 which are more robust and efficient.

3.3 Naïve Secret Sharing Based Protocol

In this section, we first briefly describe the setup of participants in the scheme, and then give a detailed description of the scheme construction in each stage.

3.3.1 Participants Setup

As described in Section 3.1.1, our scheme includes N voters, two[2] honest collectors with conflict-of-interests, and a third authority party. N voters cast their votes during the voting process, and perform verification once all votes are published. Two honest collectors also participate in every stage of the voting process, but carry different duties. They collect the initial data, the commitment, and the intermediate vote data, and then aggregate the intermediate vote data to generate all individual votes. The existence of collectors in the whole voting process deters the voters from misbehaving. The third party is not involved in any voting process but the investigation. In fact, he steps in only when a vote fails the verification. At this moment, collectors transfer all data in their hands to the third party, and he scrutinizes data to find the misbehaving voter.

The involvement of the third authority party also plays an important role of deterrence. The voters are deterred from misbehaving even from the beginning but follow the scheme rightly during the entire voting process. If all voters follow the scheme, there is no need to call in the third authority party. We assume that the third party is trusted and will not disclose voters' votes other than the misbehaving voter to either collectors or any other voters after the investigation.

3.3.2 Protocol Construction

3.3.2.1 Initialization

We assumed two collectors C_1 and C_2 for simplicity, and N registered voters V_1, V_2, $\cdots$, V_N, where the indices of both collectors and voters are known to the public, the voting registration system publishes N hash functions, H_1, H_2, $\cdots$, H_N, and a commitment scheme H. Each voter V_i applies LAS to acquire a unique position L_i where $1 \leq i \leq N$. V_i also chooses $N-1$ random numbers r_{i1}, $\cdots$, $r_{i(i-1)}$, $r_{i(i+1)}$, $\cdots$,

---

[2] As said earlier, more collectors can also be supported.

r_{iN} (Notice the index here. We do not have r_{ii}). Meanwhile, for $i < j$, when i is odd, V_i sends random numbers r_{ij} to C_1 if j is even and to C_2 otherwise, and when i is even, he sends random numbers r_{ij} to C_1 if j is odd and to C_2 if j is even. For $i > j$, when i is odd, V_i sends r_{ij} to C_1 if j is odd and to C_2 otherwise, and when i is even, he sends r_{ij} to C_1 if j is even and to C_2 if j is odd. More specifically, the random number r_{ij} goes to Collector $C_{(i+j+1) \bmod 2+1}$ for $i < j$ or to Collector $C_{(i+j) \bmod 2+1}$ for $i > j$. Consequently, each collector has exactly half set of random numbers. This prevents a collector from having all random numbers from a single voter, so the collector will not be able to deduce this voter's vote (secret). If a voter sends a number different from the one used for calculation afterwards, the receiving collector will detect it in the next stage.

3.3.2.2 Anonymization

Each voter uses (N, N)-SS to cast his vote, thus achieving anonymity. Each voter then calculates commitment of every shares, and publishes commitment data for verification by others who receive vote shares. The collectors' duty is to save the published commitment as log data for the possible investigation. This stage consists of four steps.

a) Hash Value Generation In this step, each voter V_i applies the published N hash functions against the sequence of $N-1$ random numbers, $r_{i1}, \cdots, r_{i(i-1)}, r_{i(i+1)}, \cdots, r_{iN}$, to create $N-1$ share vectors. Each vector contains N elements as follows:

$$
\begin{aligned}
E_{i1} &= [H_1(r_{i1}), H_2(r_{i1}), \cdots, H_N(r_{i1})]^T, \\
E_{i2} &= [H_1(r_{i2}), H_2(r_{i2}), \cdots, H_N(r_{i2})]^T, \\
\cdots & \\
E_{i(i-1)} &= [H_1(r_{i(i-1)}), H_2(r_{i(i-1)}), \cdots, H_N(r_{i(i-1)})]^T, \\
E_{i(i+1)} &= [H_1(r_{i(i+1)}), H_2(r_{i(i+1)}), \cdots, H_N(r_{i(i+1)})]^T, \\
\cdots & \\
E_{iN} &= [H_1(r_{iN}), H_2(r_{iN}), \cdots, H_N(r_{iN})]^T
\end{aligned}
$$

Simply put, V_i has an array of $N-1$ share vectors as $[E_{i1}, \cdots, E_{i(i-1)}, E_{i(i+1)}, \cdots, E_{iN}]$.

b) Secret Share Concretization Next, each voter V_i generates the vector E_{ii}. Assuming V_i's position L_i is j and the vote is v_i, V_i creates the ith vector E_{ii} as follows. For each element e_k ($k = 1, 2, \cdots, N$ and $k \neq j$) in the vector E_{ii}, we have:

$$e_k = -H_k(r_{i1}) - H_k(r_{i2}) - \cdots - H_k(r_{iN}) \tag{3.1}$$

And the jth element e_j in E_{ii} is:

$$e_j = v_i - H_j(r_{i1}) - H_j(r_{i2}) - \cdots - H_j(r_{iN}) \tag{3.2}$$

V_i now has an array of N share vectors as $[E_{i1}, \cdots, E_{i(i-1)}, E_{ii}, E_{i(i+1)}, \cdots, E_{iN}]$. This $N \times N$ matrix is shown in Table 3.1.

The idea is that for Equation 3.3 below

$$S_i = \sum_{l=1}^{N} E_{il} \tag{3.3}$$

where S_i is the sum of all N vectors generated by V_i, all elements in S_i are 0, except the jth element which is v_i, i.e., the vote that V_i wants to cast.

c) Commitment In the third step, V_i publishes the commitment vector C_{il} of E_{il} for $l = 1, 2, \cdots, i-1, i+1, \cdots, N$. Each element in C_{il} is a commitment of the corresponding element in E_{il}. It is unnecessary to calculate and publish the

Table 3.1.
Matrix of secret shares generated by V_i

$$\begin{pmatrix}
H_1(r_{i1}) & \cdots & H_1(r_{i(i-1)}) & -H_1(r_{i1}) - \cdots - H_1(r_{iN}) & H_1(r_{i(i+1)}) & \cdots & H_1(r_{iN}) \\
H_2(r_{i1}) & \cdots & H_2(r_{i(i-1)}) & -H_2(r_{i1}) - \cdots - H_2(r_{iN}) & H_2(r_{i(i+1)}) & \cdots & H_2(r_{iN}) \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
H_{j-1}(r_{i1}) & \cdots & H_{j-1}(r_{i(i-1)}) & -H_{j-1}(r_{i1}) - \cdots - H_{j-1}(r_{iN}) & H_{j-1}(r_{i(i+1)}) & \cdots & H_{j-1}(r_{iN}) \\
H_j(r_{i1}) & \cdots & H_j(r_{i(i-1)}) & v_i - H_j(r_{i1}) - \cdots - H_j(r_{iN}) & H_j(r_{i(i+1)}) & \cdots & H_j(r_{iN}) \\
H_{j+1}(r_{i1}) & \cdots & H_{j+1}(r_{i(i-1)}) & -H_{j+1}(r_{i1}) - \cdots - H_{j+1}(r_{iN}) & H_{j+1}(r_{i(i+1)}) & \cdots & H_{j+1}(r_{iN}) \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
H_N(r_{i1}) & \cdots & H_N(r_{i(i-1)}) & -H_N(r_{i1}) - \cdots - H_N(r_{iN}) & H_N(r_{i(i+1)}) & \cdots & H_N(r_{iN})
\end{pmatrix}$$

commitment of E_{ii} since E_{ii} is kept by V_i himself, and consequently, no one else is capable to check the commitment. Here the collectors perform the only duty in this stage by saving all voters' commitment as log data in case an investigation is required.

Meanwhile, since each collector keeps a partial set of one voter's random numbers, with both together covering the entire set, collectors can verify if the random numbers they have received from this voter is indeed the ones used in the share generation and commitment calculation. The collectors just need to use the published hash functions and commitment scheme to compute share vectors and then their commitment vectors against the received random numbers, and perform comparison between the published commitment vectors with their own.

d) Distribution In the fourth step, V_i performs $(N,\ N)$-SS by sending vectors E_{il} to V_l for $l = 1, 2, \cdots, i-1, i+1, \cdots, N$. Consequently, V_i also receives vectors E_{li} from V_l for $l = 1, 2, \cdots, i-1, i+1, \cdots, N$, with the total of $N-1$ vectors. For every vector E_{li} received, V_i verifies the vector by calculating the commitment and comparing to the one published by the sending voter V_l. If a vector fails in the verification, V_i will request V_l to resend the vector. If there is a dispute over this vector, V_i and V_l

can ask the collector, $C_{(l+i) \bmod 2+1}$, who is holding r_{li} to further verify by generating the vector and the corresponding commitment vector. We claim this misbehavior or error is correctable. If every voter follows the scheme, V_i sums up the received $N-1$ vectors together with E_{ii} to form a new vector B_i as the intermediate vote data:

$$B_i = \sum_{l=1}^{N} E_{li} \tag{3.4}$$

3.3.2.3 Collection

In this stage, each voter V_i broadcasts B_i. The collectors (voters as well, if they want to) sum up all received N vectors into a new vector Q:

$$Q = \sum_{i=1}^{N} B_i \tag{3.5}$$

The collectors publish the voting result Q for two reasons. First, two copies from different collectors can be checked for consistency, which implicitly enables mutual monitoring or even certain deterrence. Second, this allows every voter to verify his own vote. All voters can also perform this computation if they want to, and by doing so, they can even verify Q published by collectors.

3.3.2.4 Verification

If every voter follows the scheme honestly, his vote will be reflected in Q. Each voter verifies his vote in his own private position. Since he also views all other individual votes, he knows the vote total for each candidate. Thus, he can also verify if his vote is counted by checking the vote total for each candidate.

3.3.2.5 Investigation

The investigation will be initiated in the case of misbehavior. Any misbehavior before the collection stage can be in-progress detected, thus a voter cannot disturb other voters' votes during those stages. The only way that a voter can disturb others is through publishing incorrect B_i.

If a voter misbehaves in the collection stage, i.e., V_i broadcasts B_i' such that $B_i' \neq B_i$, certain individual votes in Q will not be correct. Owners of these votes will send complaint to the third authority party M_{auth}. In such circumstance, the collectors are asked to send all random numbers, together with the intermediate vote data B_i $(i = 1, 2, \cdots, N)$ and the saved log data (commitment) to M_{auth}. V_i cooperates in the investigation by giving in his position L_i, the genuine vote v_i, and the published v_i' by collectors.

M_{auth} replays the scheme to find out the misbehavior. Assuming that V_i reports his vote at the position j (i.e., L_i) is not correct, we prove[3] that the misbehaving voter can be identified without revealing any voter's position information other than V_i's position j.

Investigation - Anonymization Stage First, having all random numbers transferred from collectors, M_{auth} executes the anonymization stage.

a) Hash Value Generation: For each voter V_k where $k = 1, 2, \cdots, N$, M_{auth} runs the hash value generation step to create E_{kl} for $l = 1, 2, \cdots, k-1, k+1, \cdots, N$. However, M_{auth} is interested in the jth row (V_i's position) only since individual votes in other positions have passed verification. Therefore, M_{auth} just needs to use the hash function H_j to generate the jth element in each vector, i.e., $H_j(r_{kl})$ for $l = 1, 2, \cdots, k-1, k+1, \cdots, N$.

b) Secret Share Concretization: The jth element in E_{kk} is generated by using Equation 3.1 for $k \neq i$ since the position j is owned exclusively by V_i, not V_k, and the jth element in E_{ii} is generated by using Equation 3.2 and V_i's genuine v_i.

c) Commitment: M_{auth} so far has recovered the share vectors (To be exact, only the jth row of the matrix in Table 3.1 is needed) for each voter. The data generated in the normal scheme execution should be identical to these recovered by M_{auth}. M_{auth}

---

[3] For simplicity, our proof shown here is just for one incorrect vote. In case of multiple votes being wrong, we can apply the same technique to find all misbehaving voters since the third party has all necessary data.

verifies this by calculating the commitment of the jth element in each vector and comparing to the one given by collectors. All verification through the commitment should be passed here because any discrepancy should have already been discovered by the receiving voter in the fourth step (distribution) of the anonymization stage. If the receiving voter has missed by any chance, the misbehavior can still be detected here.

d) Distribution: M_{auth} then follows the distribution step in the anonymization stage. Having N vectors E_{kl} $(l = 1, 2, \cdots, N)$ for each voter V_k $(k = 1, 2, \cdots, N)$, M_{auth} applies Equation 3.4 to recover B_k $(k = 1, 2, \cdots, N)$. Again, M_{auth} is interested in the jth row only, so he just needs to perform computation on the jth row in each vector to generate the jth element in B_k. we now proceed to next stage.

Investigation - Collection Stage M_{auth} performs comparison to find the misbehaving voter. Having N vectors B_k $(k = 1, 2, \cdots, N)$ achieved by M_{auth} and another copy of B_k submitted by collectors, M_{auth} can single out the misbehaving voter who broadcasts the wrong B_k (or more precisely, wrong jth element in B_k).

As we have shown in the investigation, we reach a balance between anonymity/privacy (of honest voters) and traceability/accountability (of the misbehaving voter). The vote anonymity and voter privacy of all voters (even including the misbehaving voter) are preserved except the complaining voter, while we are able to trace back to the source and find the one accountable.

3.4 Security and Property Analysis

In this section, we argue that the common security requirements are satisfied. In addition, our scheme achieves double-voting detection and security deterrence.

Correctness. When all voters are honest and following the scheme, a voter V_i will have a matrix as shown in Table 3.1 after the stage in Section 3.3.2.2. The aggregation of all column vectors $[E_{i1}, \cdots, E_{i(i-1)}, E_{ii}, E_{i(i+1)}, \cdots, E_{iN}]$ in the matrix gives a vector S_i as shown in Equation 3.3. All elements in S_i are 0, except that the

jth element is V_i's vote v_i where j is the secret position V_i holds. After the collection stage in Section 3.3.2.3, we get Equation 3.5 where Q includes individual votes from all voters. Both B_i and S_i are aggregated from the same N matrices, so it is obvious that:

$$Q = \sum_{i=1}^{N} B_i = \sum_{i=1}^{N} S_i$$

Indeed Q holds all individual votes which will be verified by voters.

Vote Anonymity. By utilizing (N, N)-SS, a vote always remains anonymous if no more than $N - 2$ voters collude. Since B_i is public, collusion of $N - 1$ voters will disclose E_{ii}, and consequently, the vote as well. But collusion of up to $N - 2$ voters will not gain any information of remaining shares based on Corollary 3.2.3, thus achieving vote anonymity.

If voters collude to find the location information, the situation is slightly different. collusion of up to $N - 2$ voters will not find the exact location that each of the remaining honest voters has. However, if the votes in the remaining locations are the same, they will know how these voters have voted. We assume the majority of voters are benign. If they vote in different ways, the possibility is small for a collusion of small fraction of voters to find out how the remaining majority have voted. If the majority do vote for the same candidate, the collusion is meaningless since they cannot control the election anyway.

With the assumption of collectors being honest, although a collector has more data such as random numbers from voters thus he can further generate share vectors, the collector cannot learn a voter's vote without colluding with other voters or another collector. Thus vote anonymity is also preserved from collectors.

Verifiability. The scheme provides verifiability. First, every voter at the end of scheme can verify his own vote, thus providing individual verifiability. Second, every voter can view all other individual votes and certainly the final vote total for each candidate, thus providing universal verifiability. Furthermore, every voter (if he wants to) can participate in the collection stage to tally and verify all votes published by collectors.

Robustness. Any faulty behavior of participants can be accurately detected thus effectively preventing a voter from trying to disrupt the scheme. A misbehaving voter V_i can: (a) use a different random number in calculation other than the one sent to a collector; (b) send invalid share vector E_{ij} (where $j \neq i$) to others; (c) publish invalid commitment C_{ij} (where $j \neq i$); (d) publish invalid B_i.

We first look at Case (a). A different random number will lead to a different hash value and consequently different commitment. The collector having this random number detects this misbehavior by calculating aforementioned two values and then comparing to the commitment published in third step of the anonymization stage.

Now let's look at Case (b). A share vector is invalid when Equation 3.3 does not satisfy the condition that all elements in S_i are 0, except the jth element being V_i's vote v_i. An invalid share vector E_{ij} (where $j \neq i$) will be detected immediately by the receiving voter through checking the commitment. He will notify the sending voter if the commitment verification fails, and request a valid share vector. However, if somehow the receiving voter fails to detect the discrepancy and the voting process goes through to the end, an invalid share vector will lead into incorrect votes in certain positions. The owners of these positions will report to M_{auth} since their votes are not correctly reflected in Q. At this moment, given all data from collectors, M_{auth} will step in to replay the scheme and find out the culprit.

Case (c) will be tackled similarly as Case (b). Once a receiving voter receives a share vector from a sending voter, he will calculate the commitment using the published commitment scheme, and then compare to what the sending voter published. The receiving voter will notify the sending voter if the commitment fails, and the sending voter has to correct and re-publish the commitment. Both Case (b) and Case (c) lead to the failed verification by the receiving voter. The collector having the corresponding random number can also detect both cases, as discussed in the commitment step of the anonymization stage.

The resolution of Case (d) relies on the third authority party. Invalid B_i leads to wrong votes in Q, and consequently, the failed verification by voters. It will resort to the third authority party M_{auth} by providing him with all data from collectors. M_{auth} replays the scheme to find out the culprit.

Security Deterrence. The existence of collectors and particularly the third authority party deters voters from misbehaving and colluding. First, as discussed earlier, using a different random number will be detected by collectors. Second, collectors participate in the voting process with different duties from voter and monitor the entire voting process, which serves as a deterrence to voters. Third, the capability of the third authority party to find the misbehaving voters through the investigation also discourages voters from wrongdoing.

Fairness. The fairness is preserved through the secret sharing since any information about the votes cannot be learned until the voting results are published. First, V_i creates N share vectors with $N-1$ of them being sent to all other voters. Then each voter adds the share vectors he received from all others and the one he kept for himself, and broadcasts the intermediate vote data B_i. The collectors (and voters if they want to) finally sum up all B_i to obtain all individual votes. Until the final stage, any vote information other than his own cannot be obtained by a voter.

Vote Coercion. We argue that this scheme addresses the vote coercion issue. Not like many other schemes where all vote data is aggregated and only vote totals are published, in our scheme each individual vote is published. A voter can not only verify his own vote, but also view others' votes. A voter under coercion can certainly claim any desired vote as his and provide the position information to the coercer. The coercer can verify the vote in the given position. However, we notice the difference of handling this issue with other schemes. For example, Clarkson et al. [64] proposed to use or give away a fake credential for vote submission if under coercion. The votes by fake credentials will be identified and excluded from the final tally.

Accountability. Accountability is implemented throughout the scheme. Accountability is never an issue when all voters behave responsibly. However, if a voter tries to disrupt others' voting, we provide a mechanism to find out this voter. This is the only moment that a third authority party takes over the data to replay the scheme. As discussed in the previous section, the third party will find out the culprit.

Transparency. The scheme is transparent to voters due to the fact that all voters can not only anonymize and cast votes, but also tally votes (if they want to) of individuals and verify their own votes. Voters do not need to send their votes to the central authorities or central servers without knowing the internal functionality but participate in every stage of the voting process.

Double-voting Detection. If a voter casts vote at another position besides his own, due to (N, N)-SS addition homomorphism, the vote at this position will differ from the one casted by the owner, so the verification will fail and an investigation will then be initiated to detect the voter who casted multiple votes.

3.5 Complexity Analysis and Experiment Results

In this section, we first analyze the computation overhead, communication cost, and space complexity. Then we present results from our simulation.

3.5.1 Complexity Analysis

Here we give a theoretical analysis of the cost in computation, communication, and space at each stage. As we assumed in Section 3.3.2, there are N voters, two collectors, and one third authority party. We assume here the communication overhead of one message to be T^c.

At the initialization stage, the computation overhead of each voter generating $N-1$ random numbers is $O(NT)$. Each voter sends random numbers to collectors. This can be done with two messages, with one message containing odd-indexed numbers to one collector and another message containing even-indexed numbers to another collector, so the communication cost for each voter is $O(T^c)$. The space complexity is $O(N)$ for each voter and each collector.

The anonymization stage includes four steps. The hash value generation and the secret share concretization yield the matrix shown in Table 3.1 for V_i. The commitment produces another matrix (no computation is need for C_{ii} though) in which each element is calculated using the commitment scheme and the corresponding element in Table 3.1. Therefore, the computation overhead is $O(N^2T)$. The communication cost includes the publication of the commitment in the third step and the secret share distribution in the last step. In both steps, instead of having one element in one message, we can send one vector in one message, so the communication cost for each voter is $O(NT^c)$. The space complexity for each voter is $O(N^2)$, and for collectors is $O(N^3)$ since the commitment by each voter has to be kept for the investigation.

At the collection stage, each collector adds N vectors into one vector to obtain N individual votes, so the computation cost is $O(N^2T)$. The space cost is $O(N^2)$ since a collector needs to keep the intermediate vote data for the investigation if required.

If an authorized third party has to conduct an investigation, he will receive the random numbers, the commitment, and the intermediate vote data of each voter from collectors, and then replay the scheme. In addition, he also needs to compare both the commitment and the intermediate data achieved from his own calculation with those given by collectors. Each collector sends all required data in one message. Therefore, the communication cost of getting data from each collector is $O(T^c)$, and the computation overhead and the space complexity are $O(N^3T)$ and $O(N^3)$, respectively. We assume that collectors and, particularly, the authorized third party have more computational power and space, so they have no problem in performing the investigation.

3.5.2 Experiment Results

We present here our experiment results. Our scheme simulation is implemented in Java. All the experiments were conducted on a computer system with Intel CPU of frequency at 1.87 GHz and 16 GB memory. The Java MessageDigest class provides

MD5 and SHA algorithms. We chose MD5 as our commitment scheme. For each experiment, we ran the simulation 10 times, and then took average of the timings.

Starting with the anonymization stage, we first experimented on hash value generation and secret share concretization, which lead to the matrix in Table 3.1. The hash functions were implemented using some shift and bit-wise operations. Fig. 3.1 shows the computation time taken by one voter. The total number of voters in this figure (and the rest figures) ranges from 16 to 1400. As we discussed previously that the computation overhead is $O(N^2T)$, the curve in the figure grows as expected in polynomial of N^2 where N is the number of voters.

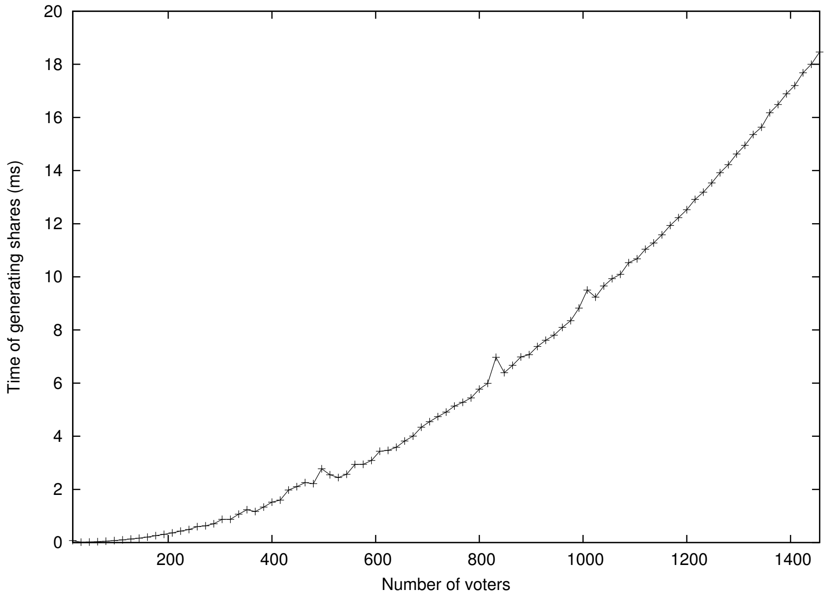


Fig. 3.1. One voter generates shares

We then performed the simulation of generating the commitment that corresponds to the third step in the anonymization stage. The commitment scheme uses the MD5 algorithm. The performance is given in Fig. 3.2. Again, the time in the figure is measured for a single voter. We can see that Figs. 3.1 and 3.2 share certain similarity, because the computation in both cases is of the same complexity, $O(N^2T)$. However, the commitment calculation takes much more time since the MD5 algorithm is quite complicated comparing to our own hash functions.

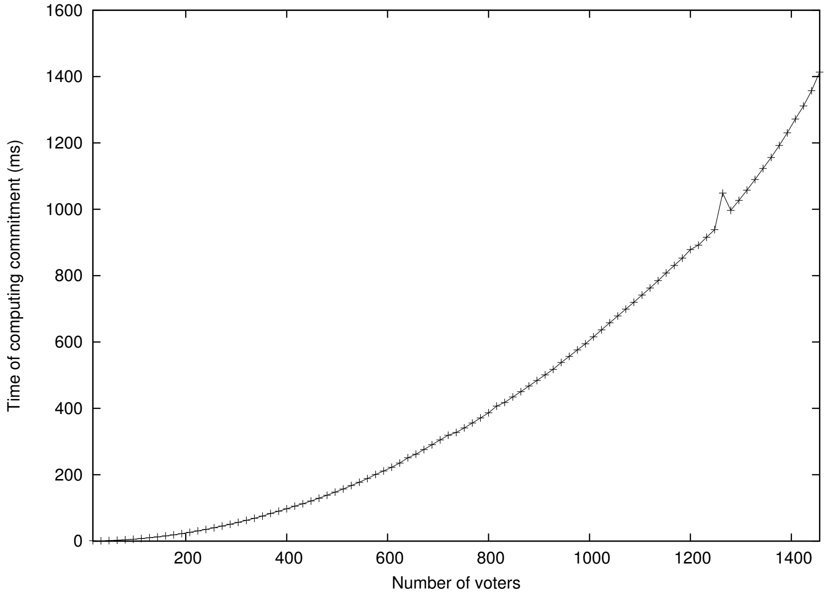


Fig. 3.2. One voter generates the commitment

We also simulated the distribution step in the anonymization stage where each voter computes B_i. Fig. 3.3 shows the performance of computation by a single voter as before. The time to compute B_i by one voter increases polynomially in N^2, which follows exactly our theoretical analysis. Comparing to the previous step, the computation here is much light.

The simulation of the collection stage is straightforward. It performs just addition operations as in the distribution step of the anonymization stage. Fig. 3.4 shows the performance of one collector aggregating all B_i into individual votes. The figure reflects the computation overhead of $O(N^2T)$ that was discussed previously.

From the simulation, we argue that our scheme is applicable to e-voting in real life together with the voting structure discussed in Chapter 9. On the other hand, we found that the commitment scheme (MD5 algorithm) is most expensive here.

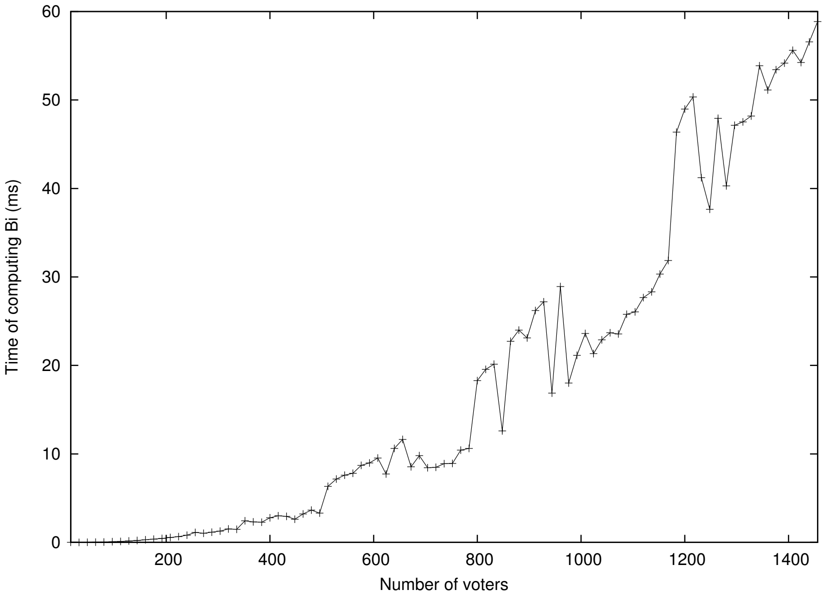


Fig. 3.3. One voter computes B_i

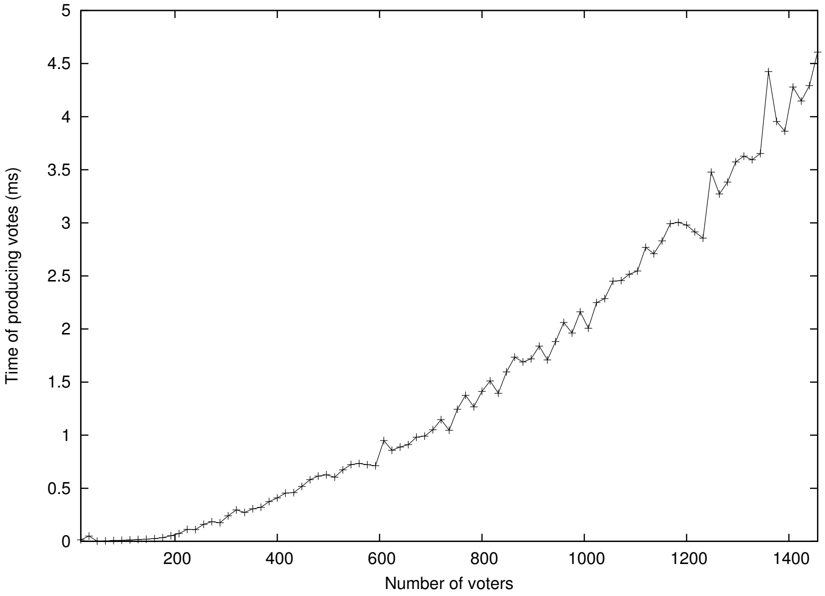


Fig. 3.4. One collector collects votes

Depending on the security requirement, we can select appropriate cryptographic hash algorithms to meet the requirement, while decreasing the overall run time.

3.6 Implementation Considerations

In this section, we discuss a few improvements that can be made in the scheme implementation to increase the efficiency and scalability.

As discussed in the scheme construction, each vote is treated independently using (N, N)-SS. With N votes, there are N independent (N, N)-SS operations. The performance will not be optimal when N is large. One improvement in the implementation is to treat a share vector E_{ij} as one single number (e.g., BigInteger in Java) and share this number using (N, N)-SS just once. Furthermore, if N is large enough that a single number cannot represent, we can split N into M sub-ranges, with each sub-range having N/M votes. For each sub-range, we use one number as mentioned earlier. Then only M independent (N, N)-SS operations are needed to share all N votes. Within each voter, M independent (N, N)-SS operations can be further parallelized using the parallel programming technique. The number of sub-ranges depends on the size of voter groups. Thus, the scheme can be effectively adapted to practical e-voting applications of different scales.

Correspondingly, The calculation of commitment can be improved as well. In the scheme, V_i commits every element in E_{ij} ($j = 1, 2, \cdots, N$ and $j \neq i$). Instead, By treating E_{ij} as a single number, V_i calculates commitment simply against this number. With the commit scheme being expensive, this implementation will dramatically improve the overall performance.

3.7 Non-interactive Voting

We can avoid the synchronization/interaction among voters during the voting process and reduce their work. We assumed that the collectors have more computational power and space. Instead of having voters generate and distribute share vectors during the voting process, we can let collectors jointly pre-generate them at the earlier stage (with each collector generating half). Suppose for a voter V_i, collectors C_1 and C_2 generate S_{i1} and S_{i2} respectively, with each element in S_{i1} and S_{i2}

are uniformly and independently selected from the domain space $\mathbb{Z}$. V_i only needs to generate E_{ii} based on his vote v_i and his position L_i using Equations 3.1 and 3.2 such that $S_i = E_{ii} + S_{i1} + S_{i2}$ (i.e., Equation 3.3). From the Theorem 1 of the INFOCOM paper [51], both E_{ii} and S_i (and certainly v_i in the S_i) are unconditionally secure. Therefore, it is safe to let C_1 and C_2 each generate half amount of random share vectors. V_i then sums up E_{ii} with other vectors received from the collectors to get B_i. V_i can publish B_i any time before the close of voting. This improvement saves the overall communication cost by avoiding the direct interaction between each pair of voters. Thus, the properties of secret sharing still hold, but voters do not need to vote at the same time. In this setup, with the assumption of collectors being honest, we may not even need to apply the commitment scheme.

In particular, the mutual restraining voting protocols in next chapter will elaborate on how to implement non-interactive voting.

4. MUTUAL RESTRAINING VOTING

In this chapter, we present our mutual restraining voting protocol [65, 66], with the interactive protocol suitable for boardroom voting, where the voting process is decentralized and the interaction between voters is encouraged, and with the non-interactive protocol for centralized voting where no interaction is needed between voters.

4.1 Protocol Overvew

We first present security assumptions and threat model in the e-voting environment, and then describe each building block of our e-voting protocol, together with high level overview of voting stages.

4.1.1 Assumptions and Attack Model

Similar to the assumptions in Section 3.1.1, there are N ($N > 4$) *voters*, V_i ($1 \leq i \leq N$), and two tallying parties, or *collectors*, C_1 and C_2[1]. But we do not have a trusted third authority party here. C_1 and C_2 *have conflicting interests*: Neither will share information with the other. The assumption of the existence of multiple conflict-of-interests collectors was previously proposed by Moran and Naor [21], and applied to real world scenarios like the two-party political system in the US. We also assume that majority of voters are benign. This assumption is reasonable since the colluding majority can easily control the election result otherwise.

In our model, collectors mutually check and restrain each other, and thus, are assumed to follow the protocol. However, unlike previous works, we do not assume that they are fully trustworthy, but only that they will not collude with each other due

---

[1] The protocol can be extended to more than two with no essential difficulties.

to conflict of interests. Our protocol ensures that neither of them can tally the ballots with the information they have before the final tallying. Some but not all voters could be malicious in our model. They can send inconsistent information to different parties or deliberately deviate from the protocol. We will show that the protocol can detect such misbehaviors and identify them without compromising honest voters' privacy.

For the interactive voting protocol presented in Section 4.2.1, *secure uncast channel* is needed between voters and between a voter and each collector. However, for the non-interactive protocol in Section 4.2.3, *secure uncast channel* between a voter and each collector is needed only. Such a secure channel can be easily provided with a public key cryptosystem.

Similar to the threat model in Section 3.1.2, we consider two types of adversaries: passive and active adversaries. The attacks can be either misbehavior from voters, collusion among voters or voters and one collector, or external attack. In this protocol, we consider the attacks targeting at voting only, rather than those targeting at general computers or network systems such a denial-of-service (DoS), DDoS, jamming, and Sybil attacks. The purposes of the attacks are either to infer a voter's vote (i.e., passive adversaries) or to change the votes which favor a particular candidate or simply invalidate the votes (i.e., active adversaries). As shown in Chapter 6, all these attacks can either be prevented or detected.

4.1.2 Cryptographic Primitives

The building blocks of our protocol include simplified (n, n)-secret sharing and tailored secure two-party multiplication. The former is given in Section 3.2.2. The latter is presented below.

4.1.2.1 Tailored Secure Two-party Multiplication (STPM)

Tailored STPM[2] is proposed in [37]. Initially, each party, M_i ($i = 1, 2$), holds a *private* input x_i. At the end of the protocol, M_i will have a *private* output r_i, such that $x_1 \times x_2 = r_1 + r_2$. The protocol works as follows:

1. M_1 chooses a private key d and a public key e for an additively homomorphic public-key encryption scheme, with encryption and decryption functions being E and D, respectively.
2. M_1 sends $E(x_1, e)$ to M_2.
3. M_2 selects a random number r_2, computes $E(x_1, e)^{x_2} E(r_2, e)^{-1}$, and sends the result back to M_1.
4. M_1 plugs the received value into D such that $r_1 = D(E(x_1, e)^{x_2} E(r_2, e)^{-1}, d)$. r_1 is M_1's private output.

4.1.3 Web Based Bulletin Board

A web based bulletin board allows anyone to monitor the dynamic vote casting and tallying in real time. Consequently, the casting and tallying processes are totally visible (i.e., transparent) to all voters. The bulletin board will dynamically display 1) on-going vote-casting, 2) incremental aggregation of the secret ballots, and 3) incremental vote counting/tallying. Note that all the incremental aggregations of secret ballots, except the final one, reveal no information of any individual vote or any candidate's count. Only at the time when the final aggregation is completed are all individual votes suddenly visible in their entirety, but in an anonymous manner. It is this sudden transition that precludes any preannouncement of partial voting results. Moreover, this transition creates a seamless connection from vote-casting

---

[2] Secure two-party multiplication (STPM) in the original paper [37] does not reveal the final product to both parties, which is different from the typical STPM where both parties know the final product. In order not to confuse readers, we rename it to tailored STPM.

and ballot confirmation to vote-tallying and verification so that both voter privacy and voter assurance can be achieved simultaneously. This is a unique feature of our voting protocol, comparing to all existing ones.

4.1.4 Technical Components

The protocol includes three important technical components (TC).

TC1: Universal verifiable voting vector. For N voters and M candidates, a voting vector $\mathbf{v}_i$ for V_i is a binary vector of $L = N \times M$ bits. The vector can be visualized as a table with N rows and M columns. Each candidate corresponds to one column. Via a robust location anonymization scheme described in Chapter 5, each voter *secretly* picks a unique row. A voter V_i will put 1 in the entry at the row and column corresponding to a candidate V_i votes for (let the position be L_c^i), and put 0 in all other entries. During the tally, all voting vectors will be aggregated. From the tallied voting vector (denoted as $\mathbf{V_A}$), the votes for candidates can be incrementally tallied. Any voter can check his vote and also visually verify that his vote is indeed counted into the final tally. Furthermore, anyone can verify the vote total for each candidate.

TC2: Forward and backward mutual lock voting. From V_i's voting vector (with a single entry of 1 and the rest of 0), a forward value v_i (where $v_i = 2^{L-L_c^i}$) and a backward value v_i' (where $v_i' = 2^{L_c^i-1}$) can be derived. Importantly, $v_i \times v_i' = 2^{L-1}$, regardless which candidate V_i votes for. During the vote-casting, V_i uses simplified (N, N)-SS to cast his vote using both v_i and v_i' respectively. v_i and v_i' jointly ensure the correctness of the vote-casting process, and enforce V_i to cast *one and only one* vote; any deviation, such as multiple voting, will be detected.

TC3: In-process check and enforcement. During the vote-casting process, collectors will jointly perform two cryptographic checks on the voting values from each voter. The first check uses tailored STPM to prevent a voter from wrongly generating his share in the vote-casting stage. The second check prevents a voter from publishing an incorrect *secret ballot* when collectors collect it from him. The secret ballot is the modular addition of a voter's own share and the share summations that the voter receives from other voters in the interactive protocol or from collectors in the non-interactive protocol.

We argue that there is no incentive for a voter to give up his own voting right and disrupt others. However, if a voter indeed puts the single 1 in another voter's location, the misbehaving voter's voting location in $\mathbf{V_A}$ and $\mathbf{V'_A}$ will be 0, leading to invalid $\mathbf{V_A}$ and $\mathbf{V'_A}$. If this happens, C_1 and C_2 can jointly find this location and then, along with the information collected during location anonymization, identify the misbehaving person.

To prevent any collector from having all $N-1$ shares of V_i, the protocol requires that C_1 have only half of V_i's shares and C_2 have the other half. Depending on whether the voting protocol is interactive or non-interactive, the arrangement of which collector getting exactly which shares is slight different as shown in Sections 4.2.1 and 4.2.3.

4.1.5 High-level Description of Protocol

Let $\mathbf{v}_i = (B_1, \ldots, B_N)$ be a voting vector. Each $B_k = (b_{k,1}, \ldots, b_{k,M})$ is a bit vector, 1 bit per candidate. To cast a ballot, a voter V_i obtains a secret and unique index L_i, which only V_i knows. To vote for the candidate c_j, V_i sets bit $b_{L_i,j}$ of B_{L_i}, and clears the other bits.

1. Voter registration. This is independent of the voting protocol. Each registered voter V_i obtains a secret index L_i using a location anonymity scheme (LAS) described in Chapter 5.
2. Voting. The voter V_i votes for the candidate c_j as follows:
    (a) Set $b_{L_i,j} = 1$ and all other bits in B_{L_i} and the other B_k, $k \neq L_i$, to 0. Call this set bit $L_c^i = (L_i - 1) \times M + j - 1$; it is simply the number of the bit when $\mathbf{v}_i$ is seen as a bit vector. See **TC1** in Section 4.1.4.

   (b) Compute $v_i = 2^{L-L_c^i}$ and $v_i' = 2^{L_c^i-1}$. This converts the bit vector to integers. Note $v_i \times v_i' = 2^{L-1}$, which is a constant. See **TC2** in Section 4.1.4.

   (c) Think of shares of all voters' ballots forming an $N \times N$ matrix (as shown in Table 4.1). Row i represents the vote v_i and Column i represents the ballot p_i. V_i computes and casts his ballot p_i as follows.

      i. In the non-interactive protocol, C_1 and C_2 generate about $\frac{N-1}{2}$ shares each for V_i. They send the sum of the shares, S_{i,C_1} and S_{i,C_2} respectively, to V_i. In the interactive protocol, V_i himself generates these $N-1$ shares. V_i then computes his own share as $s_{ii} = v_i - S_{i,C_1} - S_{i,C_2}$, which corresponds to all elements on the main diagonal of the matrix.

      ii. In the non-interactive protocol, C_1 sends voter V_i the sum $\tilde{S}_{i,C_1}$ of the shares C_1 generated for one half of the voters. Similarly, C_2 sends voter V_i the sum $\tilde{S}_{i,C_2}$ of the shares C_2 generated for the other half of the voters. In the interactive protocol, V_i receives the shares from other voters. Then V_i computes and publishes his ballot $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$.

   (d) The previous step (c) is repeated, but with v_i' instead of v_i. The share V_i obtains from this step is s_{ii}', and the ballot is p_i' which is also public.

   (e) Simultaneously with the previous step, the voter also publishes his commitments $(g^{s_{ii}}, g^{s_{ii}'}, g^{s_i s_{ii}'})$, where g is the base for the discrete logarithm problem. Two authorities jointly verify the validity of each cast ballot. See **TC3** in Section 4.1.4.

3. Tally and verification. The collectors (in fact anyone can) sum all the ballots to get $P = \sum_{i=1}^{N} p_i$ (and the corresponding $P' = \sum_{i=1}^{N} p_i'$). P and P' are public too. $\mathbf{V_A}$ and $\mathbf{V_A'}$ are in a binary form of P and P', respectively. $\mathbf{V_A}$ and $\mathbf{V_A'}$ are bit-wise identical in reverse directions. Any voter, collectors, or a third party can verify individual ballots, tallied voting vectors $\mathbf{V_A}$ and $\mathbf{V_A'}$, individual plain votes exposed in $\mathbf{V_A}$ and $\mathbf{V_A'}$, and the sum of each candidate's votes.

Fig. 4.1 shows the interaction between a voter and collectors/authorities along the voting timeline based on the interactive voting protocol that will be described next.

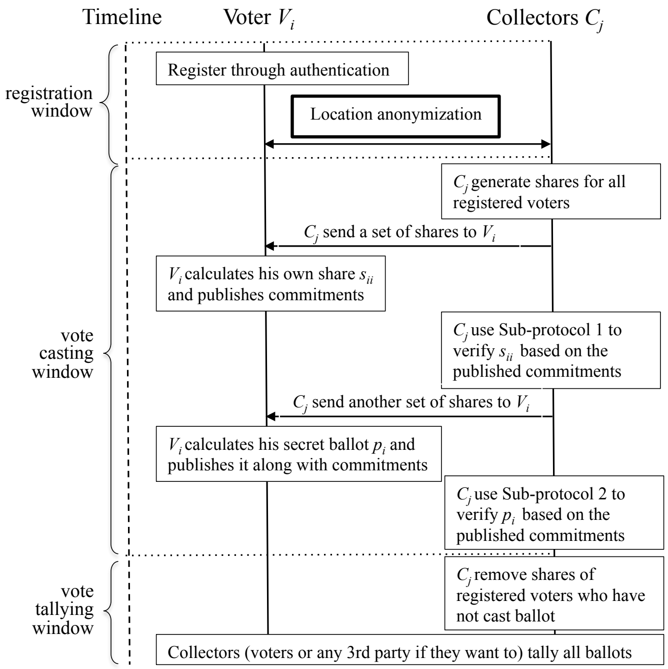

Timeline
Voter V_i
Collectors C_j
registration window
Register through authentication
Location anonymization
vote casting window
C_j generate shares for all registered voters
C_j send a set of shares to V_i
V_i calculates his own share s_{ii} and publishes commitments
C_j use Sub-protocol 1 to verify s_{ii} based on the published commitments
C_j send another set of shares to V_i
V_i calculates his secret ballot p_i and publishes it along with commitments
C_j use Sub-protocol 2 to verify p_i based on the published commitments
vote tallying window
C_j remove shares of registered voters who have not cast ballot
Collectors (voters or any 3rd party if they want to) tally all ballots

Fig. 4.1. Voter and collectors/authorities' interaction along the timeline

4.2 Mutual Restraining Voting Protocol

In this section, we elaborate on the protocol in two scenarios: interactive protocol and non-interactive protocol, together with two sub-protocols for in-process voting check and enforcement. An example of a bulletin board is then given next.

4.2.1 Interactive Voting Protocol

Stage 1: Registration (and initialization). The following computations are carried out on a cyclic group $\mathbf{Z}_{\mathbf{A}}^*$, on which the Discrete Logarithmic Problem (DLP) is intractable. $\mathbf{A} = max\{\mathbf{A}_1, \mathbf{A}_2\}$, in which $\mathbf{A}_1$ is a prime larger than 2^{1024} and $\mathbf{A}_2$ is a prime larger than $2^{2L} - 2^{L+1} + 1$. Let the number of voters be N and the number of candidates be M. The N voters are denoted as $V_1, \cdots, V_N$.

All voters have to register and be authenticated first before entering the voting system. Typical authentication schemes, such as public key authentication, can be used. Once authenticated, voter V_i executes one LAS in Chapter 5 collaboratively with other voters to obtain a unique and secret location L_i. Then V_i generates his voting vector $\mathbf{v}_i$ of the length $L = N \times M$ bits and arranges the vector into N rows (corresponding to N voters) and M columns (corresponding to M candidates); V_i fills a 1 in his row (i.e., the L_ith row) and the column for the candidate he votes, and 0 in all other entries. Consequently, the aggregation of all individual voting vectors will create a tallied vector allowing universal verifiability (**TC1**). This arrangement of vector can support voting scenarios including "yes-no" voting for one candidate and 1-out-of-M voting for M candidates with abstaining or without.

Stage 2: Vote-casting. From the voting vector $\mathbf{v}_i$ (with a singleton 1 and all other entries 0), V_i derives two decimal numbers v_i and v_i'. v_i is the decimal number corresponding to the binary string represented by $\mathbf{v}_i$, while v_i' is the decimal number corresponding to $\mathbf{v}_i$ *in reverse*.

In other words, if V_i sets the L_c^ith bit of $\mathbf{v}_i$ to 1, we have $v_i = 2^{L-L_c^i}$ and $v_i' = 2^{L_c^i-1}$, thus $v_i \times v_i' = 2^{L-1}$. v_i and v_i' are said to be *mutually restrained* (**TC2**). This feature will lead to an effective enforcement mechanism that enforces the single-voting rule with privacy guarantee: The vote given by a voter will not be disclosed as long as the voter casts one and only one vote.

Next, V_i shares v_i and v_i' with other voters using (N, N)-SS. Note that the sharing process of v_i and v_i' is independent. Assume that a secure communication channel exists between any two voters and between any voter and any collector. The following illustrates of the sharing of v_i:

1. V_i randomly selects $N-1$ shares s_{il} ($1 \leq l \leq N$, $l \neq i$) and distributes them to the other $N-1$ voters with V_l getting s_{il}. For $i < l$, when i is odd, V_i sends s_{il} to C_1 if l is even and otherwise to C_2; when i is even, V_i sends s_{il} to C_1 if l is odd

and otherwise to C_2. For $i > l$, when i is odd, V_i sends s_{il} to C_1 if l is odd and otherwise to C_2; when i is even, V_i sends s_{il} to C_1 if l is even and otherwise to C_2.That is, s_{il} goes to $C_{(i+l+1) \bmod 2+1}$ for $i < l$ or to $C_{(i+l) \bmod 2+1}$ for $i > l$. This sharing approach is similar to the one in Section 3.3.2.1. Certainly we can also use the approach shown in Table 4.1. The only objective is to prevent a single collector from obtaining enough information to infer a voter's vote. V_i then computes his own share $s_{ii} = v_i - \sum_{l=1,l\neq i}^{N} s_{il}$, and publishes the commitment $g^{s_{ii}}$.

Let the sum of shares that $C_j (j = 1, 2)$ receives from V_i be S_{i,C_j}. V_i's own share can also be denoted as: $s_{ii} = v_i - S_{i,C_1} - S_{i,C_2}$.

2. Upon receiving $N - 1$ shares from other voters, V_i computes the secret ballot, $p_i = \sum_{l=1}^{N} s_{li}$, which is the sum of the received $N - 1$ shares and his own share s_{ii}, and then broadcasts p_i.

   Two collectors also have these $N - 1$ shares, with each having a subset. Let the sum of the subset of shares held by the collector $C_j (j = 1, 2)$ be $\tilde{S}_{i,C_j}$. The secret ballot can also be denoted as: $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$.

The sharing of v'_i is the same as above, and $g^{s'_{ii}}$ is also published during this process. In addition, V_i publishes $g^{s_{ii}s'_{ii}}$. These commitments, $g^{s_{ii}}, g^{s'_{ii}}$, and $g^{s_{ii}s'_{ii}}$, are used by the collectors to enforce that a voter generates and casts his vote by distributing authentic shares and publishing an authentic secret ballot p_i (i.e., the two sub-protocols described in Section 4.2.2).

Stage 3: Collection/Tally. Collectors, and voters if they want to, collect secret ballots p_i $(1 \leq i \leq N)$ from all voters and obtain $P = \sum_{i=1}^{N} p_i$. P is decoded into a tallied binary voting vector $\mathbf{V_A}$ of length L. The same is done for p'_i $(1 \leq i \leq N)$ to obtain P', and consequently $\mathbf{V'_A}$. If voters have followed the protocol, these two vectors should be reverse to each other by their initialization in Stage 1.

It might be possible that some voters do not cast their votes, purposely or not, which can prevent $\mathbf{V_A}$ or $\mathbf{V'_A}$ from being computed. If V_i's ballot does not appear on the bulletin board after the close of voting, all shares V_i sent to and received from other voters have to be canceled out from P. Since $s_{ii} = v_i - S_{i,C_1} - S_{i,C_2}$ and $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$, we have:

$$p_i = v_i - S_{i,C_1} - S_{i,C_2} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$$

Without V_i casting his secret ballot, we simply deduct $(S_{i,C_1} + S_{i,C_2} - \tilde{S}_{i,C_1} - \tilde{S}_{i,C_2})$ from P. Similar deduction also applies to P'.

Stage 4: Verification. Anyone can verify whether $\mathbf{V_A}$ is a reverse of $\mathbf{V'_A}$ and whether each voter has cast one and only one vote. V_i can verify the entry L^i_c (corresponding to the candidate that V_i votes for) has been correctly set to 1 and the entries for other candidates are 0. Furthermore, the tallied votes for all candidates can be computed and verified via $\mathbf{V_A}$ and $\mathbf{V'_A}$. In summary, both individual and universal verification are naturally supported by this protocol.

4.2.2 Two Sub-protocols

A voter may misbehave in different ways. Examples include: 1) multiple voting; 2) disturbing others' voting; and 3) disturbing the total tally. All examples of misbehavior are equivalent to an offender inserting multiple 1s in the voting vector. The following two sub-protocols, which are collectively known as *in-process check and enforcement* (**TC3**), ensure that each voter should put a single 1 in his voting vector, i.e., vote once and only once.

4.2.2.1 Sub-protocol 1

1) Recall that in Stage 2, V_i sends $N-1$ shares s_{il} $(1 \leq l \leq N, l \neq i)$ of his secret vote v_i to the other $N-1$ voters as well as the two collectors; each of the two collectors, C_1 and C_2, has a subset of the $N-1$ shares denoted as S_{i,C_1} and S_{i,C_2} respectively. Similarly for v'_i, C_j $(j = 1, 2)$ gets S'_{i,C_j}.

2) Since V_i has published $g^{s_{ii}}$ and $g^{s'_{ii}}$, C_1 can compute $(g^{s_{ii}})^{S'_{i,C_1}}$ and $(g^{s'_{ii}})^{S_{i,C_1}}$. In addition, C_1 computes $g^{S_{i,C_1}S'_{i,C_1}}$. Similarly, C_2 computes $(g^{s_{ii}})^{S'_{i,C_2}}$, $(g^{s'_{ii}})^{S_{i,C_2}}$, and $g^{S_{i,C_2}S'_{i,C_2}}$.

3) C_1 and C_2 cooperatively compute $g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$. A straightforward application of Diffie-Hellman key agreement [67] to obtain $g^{S_{i,C_1}S'_{i,C_2}}$ and $g^{S'_{i,C_1}S_{i,C_2}}$ will not work.[3] Hence, tailored STPM is used to compute $g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$ without disclosing $g^{S_{i,C_1}}, g^{S'_{i,C_1}}, g^{S_{i,C_2}}$ and $g^{S'_{i,C_2}}$ as follows:

- Execute the tailored STPM, C_1 and C_2 obtain r_1 and r'_2 respectively such that $r_1 + r'_2 = S_{i,C_1}S'_{i,C_2}$.

- Execute the tailored STPM, C_1 and C_2 obtain r'_1 and r_2 respectively such that $r'_1 + r_2 = S'_{i,C_1}S_{i,C_2}$. (Exchanging r_1 and r'_2 or r'_1 and r_2 between C_1 and C_2 will not work.[4])

- C_1 computes $g^{r_1+r'_1}$, C_2 computes $g^{r_2+r'_2}$. Obviously we have: [5]
  $g^{r_1+r'_1} \times g^{r_2+r'_2} = g^{r_1+r'_2+r'_1+r_2} = g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$.

4) C_1 computes a combined product $K_1 = (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times g^{r_1+r'_1}$ and similarly, C_2 computes a combined product $K_2 = (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_2+r'_2}$, and then they exchange K_1 and K_2.

5) Using V_i's commitment $g^{s_{ii}s'_{ii}}$ and K_1, K_2, each collector obtains $g^{s_{ii}s'_{ii}} \times K_1 \times K_2 = g^{s_{ii}s'_{ii}} \times (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$. The collectors can verify that the product equals $g^{2^{L-1}}$. If not, V_i must have shared v_i and/or v'_i incorrectly.

---

[3] If C_1 exchanges his $g^{S_{i,C_1}}$ and $g^{S'_{i,C_1}}$ with C_2's $g^{S_{i,C_2}}$ and $g^{S'_{i,C_2}}$, since $g^{s_{ii}}$ and $g^{s'_{ii}}$ are published by V_i, C_1 and C_2 each can obtain $g^{s_{ii}+S_{i,C_1}+S_{i,C_2}}$ and $g^{s'_{ii}+S'_{i,C_1}+S'_{i,C_2}}$ which correspond to g^{v_i} and $g^{v'_i}$. Because there are only L possibilities of each voter's vote, C_1 and C_2 each can simply try L values to find out the vote v_i. This violates voter privacy since both the vote and the location are known.

[4] If C_1 and C_2 exchange r_1 and r'_2 for an example, C_2 can obtain $g^{S_{i,C_1}}$ since he has r'_2 and S'_{i,C_2}. With $g^{s_{ii}}$ being public, and $v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}$, C_2 can get g^{v_i} by trying out L values and consequently find out the vote v_i. Likewise, C_1 can also find out v_i.

[5] C_1 and C_2 cannot exchange $g^{r_1+r'_1}$ and $g^{r_2+r'_2}$ directly. Doing so will result in a brute-force attack which is able to obtain the voter's vote, as described in Section 6.1.

4.2.2.2 Sub-protocol 2

While the Sub-protocol 1 ensures that V_i should generate s_{ii} and all shares properly, Sub-protocol 2 enforces that V_i should faithfully publish the secret ballots, p_i and p'_i.

1) Recall that in the sharing of v_i, V_i receives $N - 1$ shares from other voters, and these shares are also received by collectors. Each of the collectors, C_1 and C_2, receives a subset of these $N - 1$ shares, so trust is split between two collectors. The sum of the subset of shares held by the collector $C_j (j = 1, 2)$ is $\tilde{S}_{i,C_j}$. C_j $(j = 1, 2)$ will exchange $g^{\tilde{S}_{i,C_j}}$. Similarly for v'_i, C_j $(j = 1, 2)$ will exchange $g^{\tilde{S}'_{i,C_j}}$.

2) From the published p_i and p'_i, the collectors compute g^{p_i} and $g^{p'_i}$. Since $g^{s_{ii}}$ and $g^{s'_{ii}}$ are published and verified in Sub-protocol 1, collectors will verify that $g^{s_{ii}} g^{\tilde{S}_{i,C_1}} g^{\tilde{S}_{i,C_2}} = g^{p_i}$ and $g^{s'_{ii}} g^{\tilde{S}'_{i,C_1}} g^{\tilde{S}'_{i,C_2}} = g^{p'_i}$. If either of these fails, V_i must have published the wrong secret ballots p_i and/or p'_i.

4.2.3 Non-interactive Voting Protocol

The protocol presented in Section 4.2.1 is suitable for voting scenarios such as boardroom voting where voters are encouraged to interact with each other. However, it is often the case that an election involves a large group of people where the interaction is impossible to be realistic. In this scenario, we allow collectors to carry the duty of creating voters' shares. While this eliminates the interaction between voters, the properties of our voting protocol remain held as we will discuss in the next section.

As said earlier, it is not practical to require a large number of voters to interact with each other during voting. Fortunately, our (N, N)-SS based voting protocol can be designed to allow voters to vote non-interactively. Table 4.1 illustrates how the two collectors generate respective shares for voters. In this non-interactive vote-casting, two collectors generate shares for every voter, and interact with a voter for vote-casting whenever the voter logs into the system to vote. Compared to the interactive

Table 4.1.
Collectors generate all shares for each voter, with each collector generating about half of $N-1$ shares

$$
\begin{array}{cc|cccc|ccc|c}
 & & \leftarrow - & & C_1 & - \rightarrow & \leftarrow - & C_2 & - \rightarrow & \\
 & & & & \tilde{S}_{i,C_1} & & & \tilde{S}_{i,C_2} & & \\
 & & & & \Downarrow & & & \Downarrow & & \\
\hline
 & v_1 & s_{1,1} & s_{\widehat{1,2}} & \cdots & s_{\widehat{1,N/2}} & s_{\check{1,N/2+1}} & \cdots & s_{\check{1,N}} & \\
C_1,\ S_{i,C_1} \Rightarrow & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \Leftarrow S_{i,C_2},\ C_2 \\
 & v_{N/2} & s_{\widehat{N/2,1}} & s_{\widehat{N/2,2}} & \cdots & s_{N/2,N/2} & s_{\check{N/2,N/2+1}} & \cdots & s_{\check{N/2,N}} & \\
\hline
 & v_{N/2+1} & s_{\check{N/2+1,1}} & s_{\check{N/2+1,2}} & \cdots & s_{\check{N/2+1,N/2}} & s_{N/2+1,N/2+1} & \cdots & s_{\widehat{N/2+1,N}} & \\
C_2,\ S_{i,C_2} \Rightarrow & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \Leftarrow S_{i,C_1},\ C_1 \\
 & v_N & s_{\check{N,1}} & s_{\check{N,2}} & \cdots & s_{\check{N,N/2}} & s_{\widehat{N,N/2+1}} & \cdots & s_{N,N} & \\
\hline
 & & & & \Uparrow & & & \Uparrow & & \\
 & & & & \tilde{S}_{i,C_2} & & & \tilde{S}_{i,C_1} & & \\
 & & \leftarrow - & & C_2 & - \rightarrow & \leftarrow - & C_1 & - \rightarrow & \\
\end{array}
$$

Notice: v_i is V_i's vote. $s_{\widehat{i,j}}$ $(i \neq j)$ is generated by C_1, and $s_{\check{i,j}}$ $(i \neq j)$ is by C_2. S_{i,C_1} is the sum of all shares ($s_{\widehat{i,j}}$) in Row i generated by C_1 for V_i, and S_{i,C_2} is the sum of all shares ($s_{\check{i,j}}$) in Row i by C_2 for V_i. $s_{i,i} = v_i - S_{i,C_1} - S_{i,C_2}$. $\tilde{S}_{i,C_1}$ is the sum of all shares ($s_{\widehat{j,i}}$) in Column i generated by C_1, and $\tilde{S}_{i,C_2}$ is the sum of all shares ($s_{\check{j,i}}$) in Column i by C_2.

protocol, only **Stage 2** is different, while the rest of the stages are the same. Steps in this new **Stage 2** are given below.

- The two collectors work together to generate all $N-1$ shares for each voter V_i in advance as shown in Table 4.1, with $s_{\widehat{i,j}}$ $(i \neq j)$ by C_1 and $s_{\check{i,j}}$ $(i \neq j)$ by C_2. s_{ii} is derived by V_i himself. Specifically[6], for the voters V_1 to $V_{N/2}$, C_1 generates their first $N/2-1$ shares (up-left of the matrix) and C_2 generates their last $N/2$ shares (up-right of the matrix). For the voters $V_{N/2+1}$ to V_N, C_1 generates their last $N/2-1$ shares (lower-right of the matrix) and C_2 generates their first $N/2$ shares (lower-left of the matrix). Fig. 4.2 illustrates a case of five voters.

---

[6] $N/2$ should be in the form of $\lceil N/2 \rceil$ by applying the ceiling function (or $\lfloor N/2 \rfloor$ by taking the floor function). $(N-1)/2$ afterwards should also be in the same form. We omit this notation simply for the sake of readability.

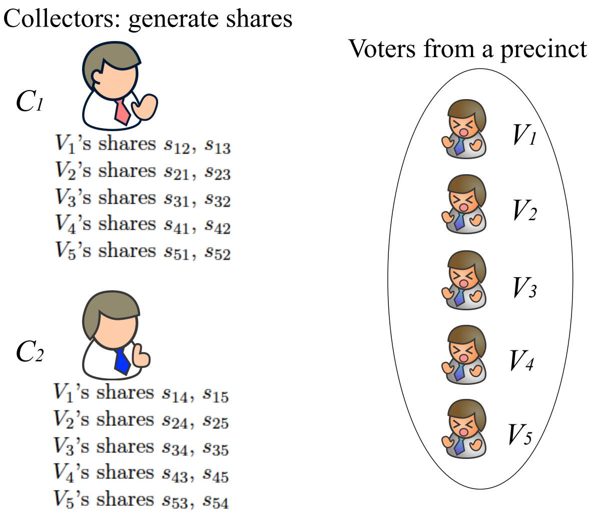
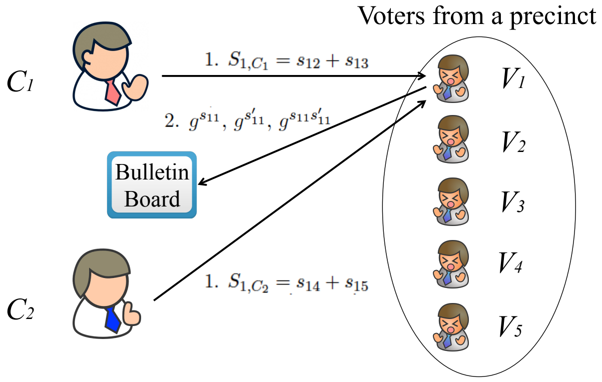

Collectors: generate shares
Voters from a precinct
C_1
V_1's shares s_{12}, s_{13}
V_2's shares s_{21}, s_{23}
V_3's shares s_{31}, s_{32}
V_4's shares s_{41}, s_{42}
V_5's shares s_{51}, s_{52}
C_2
V_1's shares s_{14}, s_{15}
V_2's shares s_{24}, s_{25}
V_3's shares s_{34}, s_{35}
V_4's shares s_{43}, s_{45}
V_5's shares s_{53}, s_{54}
V_1
V_2
V_3
V_4
V_5

Fig. 4.2. Collectors generate shares for voters

- Whenever a voter V_i logs into the system to cast his vote, the two collectors will each send their half of $N-1$ shares (in fact, the sum of these shares, denoted as S_{i,C_j} in Table 4.1, where $j = 1$ or 2) to this voter. Specifically, C_1 sends S_{i,C_1} (the sum of shares in one half of the ith row) to the voter, and C_2 sends S_{i,C_2} (the sum of shares in the other half of the ith row) to the voter. The voter V_i will compute his own share as $s_{ii} = v_i - S_{i,C_1} - S_{i,C_2}$, and publish his commitments (i.e., $g^{s_{ii}}$, $g^{s'_{ii}}$, $g^{s_{ii}s'_{ii}}$). Fig. 4.3 shows the communication between collectors and voters. Under the assumption that the two collectors have conflicting interests, they will not give away a voter's shares to each other, so neither of them can derive the voter's vote from V_i's commitment.

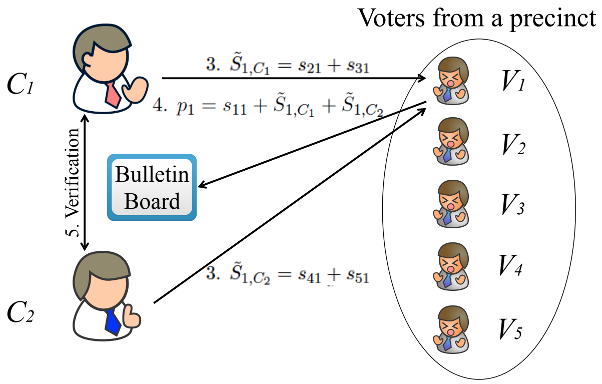
- The two collectors verify a voter's vote using **Sub-protocol 1** and if passed, send the shares from the other $N-1$ voters (one from each voter) to this voter. Specifically, C_1 sends $\tilde{S}_{i,C_1}$ (the sum of shares in one half of the ith column as shown in Table 4.1) to the voter, and similarly C_2 sends $\tilde{S}_{i,C_2}$ (the sum of shares in the other half of the ith column). The voter sums the shares from the two collectors and his own share, and then publishes the secret ballot of $s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$ to the bulletin board, as shown in Fig. 4.4 as an example.


Voters from a precinct
C_1
1. $S_{1,C_1} = s_{12} + s_{13}$
2. $g^{s_{11}}$, $g^{s'_{11}}$, $g^{s_{11}s'_{11}}$
Bulletin Board
C_2
1. $S_{1,C_2} = s_{14} + s_{15}$
V_1
V_2
V_3
V_4
V_5

1. When V_1 logs into the system to cast his vote, C_j $(j = 1, 2)$ will send V_1 the sum of shares (S_{1,C_j}) C_j generates for V_1
2. V_1 computes his own share $s_{11} = v_1 - S_{1,C_1} - S_{1,C_2}$,

Fig. 4.3. Voters interact with collectors to calculate s_{ii} and commitments

(Optionally, the voter can send to just two collectors, or to only one of the collectors to prevent the collector initiated voter coercion.) The two collectors can verify the voter's ballot using **Sub-protocol 2**.


Voters from a precinct
C_1
3. $\tilde{S}_{1,C_1} = s_{21} + s_{31}$
4. $p_1 = s_{11} + \tilde{S}_{1,C_1} + \tilde{S}_{1,C_2}$
5. Verification
Bulletin Board
C_2
3. $\tilde{S}_{1,C_2} = s_{41} + s_{51}$
V_1
V_2
V_3
V_4
V_5

3. C_j $(j = 1, 2)$ sends V_1 the sum of shares $(\tilde{S}_{1,C_j})$ he generated for other voters to V_1
4. V_1 computes and publishes $p_1 = s_{11} + \tilde{S}_{1,C_1} + \tilde{S}_{1,C_2}$
5. Two collectors together run sub-protocols for verification

Fig. 4.4. Voters interact with collectors to compute secret ballots

It is clear that although the two collectors generate shares for a voter, neither of them can obtain the voter's own share s_{ii} or the voter's vote v_i, unless two collectors collude and exchange the shares they have generated. As proven by Theorem 3.2.1, any k voters, as long as $k \leq N - 1$, can not obtain the share (thus, the vote) of any other voters in an unconditionally secure manner. Again, as in the interactive protocol, it may be possible that some voters do not cast their votes, preventing $\mathbf{V_A}$ from being computed. The solution discussed in Stage 3 of Section 4.2.1 still applies.

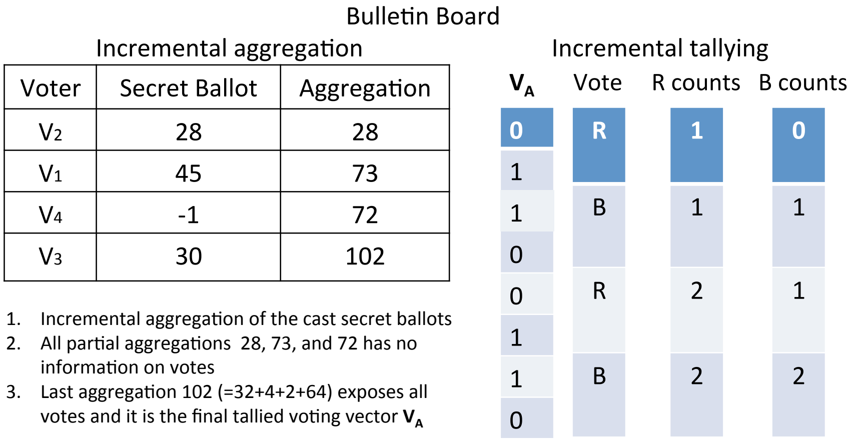
4.2.4 One Example of Web Based Bulletin Board

As discussed earlier, our web based bulletin board displays the on-going vote casting and tallying processes. The incremental aggregation of secret ballots does not reveal information about any individual vote. Only when the final aggregation is completed, all individual votes in the voting vector are suddenly visible in their entirety to the public, but in an anonymous manner. It is this sudden transition that precludes preannouncement of any partial voting results.

Table 4.2.
A voting example involving 4 voters and 2 candidates (R and B)

Voter	Location	Vote	Shares	Secret ballot
V_1	2	B (32)	$\underline{12}$+5+8+7	45 (=12+1+15+17)
V_2	3	R (4)	1+$\underline{13}$+(-3)+(-7)	28 (=5+13+7+3)
V_3	4	B (2)	15+7+($\underline{-10}$)+(-10)	30 (=8+(-3)+(-10)+35)
V_4	1	R (64)	17+3+35+$\underline{9}$	-1 (=7+(-7)+(-10)+9)

For the illustration purpose only, Table 4.2 gives an example of 4 voters with their corresponding shares and secret ballot in a case of 2 candidates. Fig. 4.5 presents the aggregation and tallying on the bulletin board. It is obvious that the incremental aggregation does not disclose any information until the last secret ballot, V_3's 30, is counted in.



Bulletin Board

Incremental aggregation

Voter	Secret Ballot	Aggregation
V_2	28	28
V_1	45	73
V_4	-1	72
V_3	30	102

1. Incremental aggregation of the cast secret ballots
2. All partial aggregations 28, 73, and 72 has no information on votes
3. Last aggregation 102 (=32+4+2+64) exposes all votes and it is the final tallied voting vector $\mathbf{V_A}$

Incremental tallying

$\mathbf{V_A}$	Vote	R counts	B counts
0	R	1	0
1			
1	B	1	1
0			
0	R	2	1
1			
1	B	2	2
0			

Fig. 4.5. Real-time bulletin board

4.3 Complexity Analysis and Simulation

In this section, we analyze the complexity of our mutual restraining protocol including both interactive voting and non-interactive voting, and then give simulation results.

4.3.1 Performance and Complexity Analysis

Here we analyze the computational complexity and communication cost for both voters and collectors in the protocol. Suppose that each message takes T bits. Since the protocol works on a cyclic group $\mathbf{Z}^*_{\mathbf{A}}$ ($\mathbf{A} = max\{\mathbf{A}_1, \mathbf{A}_2\}$, in which $\mathbf{A}_1$ is a prime greater than 2^{1024} and $\mathbf{A}_2$ is a prime greater than $2^{2L} - 2^{L+1} + 1$), we see that $T = O(L)$.

The voting protocol involves two independent sharing processes of v_i and v'_i. Each voter's communication cost is calculated as follows. In the interactive protocol, each voter sends shares of v_i to the other $N-1$ voters and the two collectors, which costs $O(NT)$. In the non-interactive protocol however, each voter receives shares from the collectors only, so the cost is $O(T)$. In both protocols, each voter also publishes p_i

and the commitments $g^{s_{ii}}$, $g^{s'_{ii}}$, and $g^{s_{ii}s'_{ii}}$, which costs $O(T)$. Therefore, the total communication cost of sharing of v_i for a voter is $O(NT) + O(T)$ in the interactive protocol and $O(T)$ in the non-interactive protocol. The communication cost of sharing v'_i is the same.

Each voter's computation cost includes computing v_i, generating N shares (in the interactive protocol only), computing the secret ballot p_i, and computing the commitments $g^{s_{ii}}$, $g^{s'_{ii}}$, and $g^{s_{ii}s'_{ii}}$, each of which costs $O(T)$, $O(NT)$ (in the interactive protocol only), $O(NT)$ in the interactive protocol and $O(T)$ in the non-interactive protocol, and $O(T^3)$ respectively. The same computation cost applies to the sharing of v'_i. **Notes**: The commitments can typically be computed by a calculator efficiently, thus, the complexity of $O(T^3)$ will not become a performance issue.

The collector C_j's communication cost involves: 1) receiving $O(N)$ shares from each voter in the interactive protocol with the cost of $O(NT)$, or sending sums of shares to each voter in the non-interactive protocol with the cost of $O(T)$; 2) exchanging data with the other collector in Sub-protocol 1 with the cost of $O(\tilde{T})$ (assuming that the tailored STPM messages are encoded into $\tilde{T}$-bits); and 3) publishing $g^{\tilde{S}_{i,C_j}}$ or $g^{\tilde{S}'_{i,C_j}}$ for each voter in Sub-protocol 2 with the cost of $O(T)$. With N voters, the total cost for each collector is $O(NT) + O(\tilde{T}) + O(T))N$ for the interactive protocol and $(O(T) + O(\tilde{T}) + O(T))N$ for the non-interactive protocol.

The computation cost of each collector includes generating $N(N-1)/2$ shares for all voters (in the non-interactive protocol only) which costs $O(N^2T)$, summing up the p_i during voting collection/tally, which costs $O(NT)$, and running Sub-protocol 1 and Sub-protocol 2 whose computation costs are shown below.

In Sub-protocol 1, for the collector C_j, 1) computing $g^{s_{ii}S'_{i,C_j}}$, $g^{s'_{ii}S_{i,C_j}}$ and $g^{S_{i,C_j}S'_{i,C_j}}$ costs $O(T^3)$; 2) computing K_j involves the tailored STPM; and 3) computing $g^{s_{ii}s'_{ii}} \times K_1 \times K_2$ costs $O(T^2)$. Computing K_j consists of obtaining r_j and r'_j with the tailored STPM, computing $g^{r_j+r'_j}$ and multiplying this with other terms. Let the complexity for tailored STPM be $O(TPMC)$. The total computation cost of Sub-protocol 1 for each collector is $O(T^3) + O(T^2) + O(TPMC)$ per voter.

In Sub-protocol 2, the collectors: 1) compute $\tilde{S}_{i,C_j}$ and $\tilde{S}'_{i,C_j}$; 2) compute $g^{\tilde{S}_{i,C_j}}$ and $g^{\tilde{S}'_{i,C_j}}$; 3) multiply $g^{s_{ii}}$, $g^{\tilde{S}_{i,C_1}}$ and $g^{\tilde{S}_{i,C_2}}$, and also $g^{s'_{ii}}$, $g^{\tilde{S}'_{i,C_1}}$ and $g^{\tilde{S}'_{i,C_2}}$; and 4) compute g^{p_i} and $g^{p'_i}$. These computations cost $O(NT)$, $O(T^3)$, $O(T^2)$ and $O(T^3)$, respectively. Thus, the total computation cost of Sub-protocol 2 is $O(NT)+O(T^3)+O(T^2)+O(T^3)$ for each voter.

4.3.2 Simulation Results

The results presented here are from our protocol simulation implemented in Java. The experiments were carried out on the same machine that we did for the protocol in Chapter 3. Again for each experiment, we took the average of 10 rounds of simulation. 1-out-of-2 voting is simulated. Thus, the length of the voting vector is $L = 2N$ where N is the number of voters.

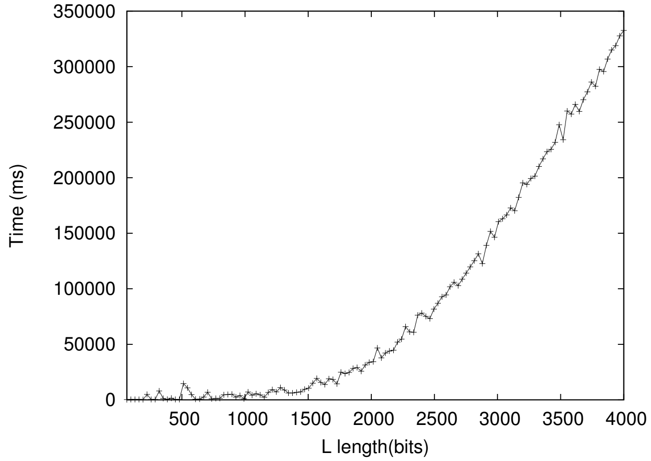


Fig. 4.6. Collectors run Sub-protocol 1 in TC3 against one voter

In the non-interactive protocol, the computation time for a voter V_i is negligible since only two subtractions are needed for s_{ii} and two additions for p_i, and the commitments can be obtained by using a calculator sufficiently. Collectors however require heavy load of calculation, so our simulation focuses on collectors' operations.

Figs. 4.6 and 4.7 show the computation time of Sub-protocol 1 and Sub-protocol 2, respectively. Sub-protocol 1 is dominated by the tailored STPM, due to the com-

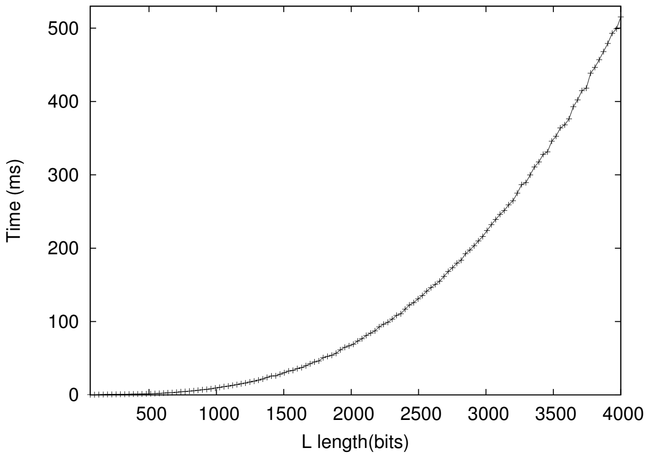


Fig. 4.7. Collectors run Sub-protocol 2 in TC3 against one voter

putationally intensive Paillier Cryptosystem used in our implementation. However, this should not be an issue in real life since the collectors usually possess much greater computing power.

Fig. 4.8 shows the time for one collector to collect and tally votes. The execution time depends on the number of voters N and the length L. As L increases, the voting collection/tally time increases by $NL = O(L^2)$.

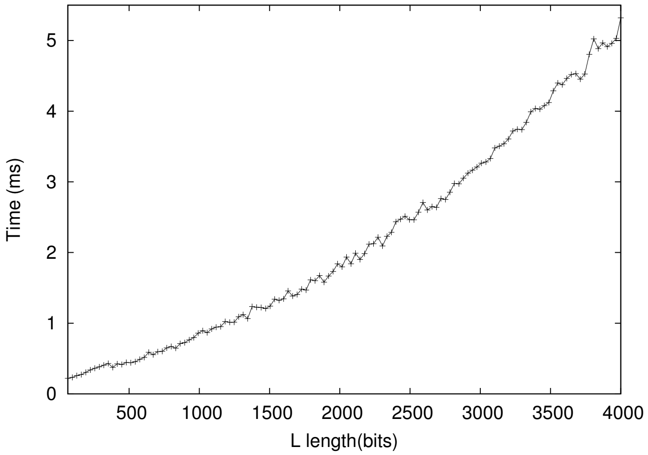


Fig. 4.8. One collector collects/tallies votes

The simulation results confirm the performance analysis in Section 4.3.1. Most operations are quite efficient. For example, when $L = 4000$ (and $N = 2000$), collecting and tallying votes took only 0.005 seconds. For the in-process enforcement protocol however, it took the collectors 332 seconds to complete Sub-protocol 1 and 0.515 seconds to complete Sub-protocol 2. To amortize the relatively high cost, the collectors may randomly sample voters for misbehavior checking and only resort to full checking when a discrepancy in the tally is detected or reported.

5. DESIGN OF ROBUST AND EFFICIENT LOCATION ANONYMIZATION SCHEMES

Location anonymization scheme (LAS) is a key component of our voting protocols. Inspired by the work in [61], we propose a new set of LAS that is robust and efficient. Our new schemes solve the following problem with the previous scheme: If a member misbehaves in next rounds by selecting multiple locations or a location that is already occupied by another member, the location selection in [61] may never finish.

5.1 Original Location Anonymization Scheme

Our first LAS is based on the mutual lock voting mechanism in Chapter 4. It works as follows:

1. Each voter V_i initializes a location vector $\mathbf{L}_i$ (of length $\bar{L}$) with 0s. V_i randomly selects a location $\hat{L}_i$ ($1 \leq \hat{L}_i \leq \bar{L}$) and sets the $\hat{L}_i$th element/bit of $\mathbf{L}_i$ to 1.
2. From $\mathbf{L}_i$, V_i obtains two values l_i and l'_i by: 1) encoding $\mathbf{L}_i$ into a decimal number l_i[1]; and 2) reversing $\mathbf{L}_i$ to be $\mathbf{L}'_i$ and encoding it into a decimal number l'_i. For example, if $\mathbf{L}_i = [000010]$, we obtain $l_i = 10$ and $l'_i = 10000$. Evidently, $l_i \times l'_i = 10^{\bar{L}-1}$.
3. V_i shares l_i and l'_i using (N, N)-SS as in Stage 2 in Section 4.2.1. All voters can obtain the aggregated location vector $\mathbf{L}_A$ and $\mathbf{L}'_A$. If V_i has followed the protocol, $\mathbf{L}_A$ and $\mathbf{L}'_A$ are the reverse of the other.

---

[1] A decimal encoding, instead of a binary one, is used to encode $\mathbf{L}_i$. The motivation is illustrated below. Assume that the binary encoding is adopted. Let the location vectors of voters V_i, V_j and V_k be $\mathbf{L}_i = 000010$, $\mathbf{L}_j = 000010$, and $\mathbf{L}_k = 000100$, respectively. Therefore, $\mathbf{L}_A = 001000$: Voters cannot tell if they have obtained unique locations. This will not be the case if $\mathbf{L}_i$ uses a larger base. However, encoding $\mathbf{L}_i$ in a larger base consumes more resources. Decimal is a trade-off we adopted to strike a balance between fault tolerance and performance. The probability of having more than 10 voters collide at the same location is considerably lower than that of 2.

4. V_i checks if the $\hat{L}_i$th element/bit of $\mathbf{L}_A$ is 1. If so, V_i has successfully selected a location without colliding with others. V_i also checks if everyone has picked a non-colliding location by examining whether $max(\mathbf{L}_A) = 1$ and whether the total number of 1s equals to the number of voters. If there is at least one collision, steps 1 through 3 will restart. In a new round, voters who have successfully picked a location without collision in the previous round keep the same location, while others randomly select from locations not been chosen.

5. The in-process check and enforcement mechanism in Section 4.2.2 is concurrently executed by collectors to enforce that a voter will select one and only one location in each round. Furthermore, the mechanism, to be proved in Section 6.7.1, ensures that any attempt of inducing collision by deliberately selecting an occupied position will be detected. Hence, such misbehavior will be precluded.

6. Once all location collisions are resolved in a round, each voter removes non-occupied locations ahead of his own and obtains his real location $L_i = \sum_{j=1}^{\hat{L}_i} (\mathbf{L}_A)_j$. After the adjustment, the occupied locations become contiguous. The length of the adjusted $\mathbf{L}_i$ equals to the number of voters, N.

We will complement the above discussion with simulation result in Section 5.4 and security analysis in Section 6.7.1.

Notes: 1) Location anonymization, a special component in our protocols, seems to be an additional effort for voters. However, it is beneficial since voters not only select their secret locations, but also learn/practice vote-casting ahead of the real election. The experiments show that 2 to 3 rounds are generally enough. 2) Location anonymization can be executed non-interactively. 3) A malicious participant deliberately inducing a collision by choosing an already occupied location will be identified.

Under the assumption that C_1 and C_2 have conflicting interests and thus will check each other but not collude, more deterministic and efficient LAS can be designed. One algorithm can be: two collectors perform double encryption (of 1 to N) and double shuffle before sending results to voters in a way such that neither can determine which voter gets which number, even though a collector may collude with some voter(s).

The LAS elaborated above may need multiple rounds, which can be relatively inefficient and inconvenient to voters. To solve this problem, we propose two new LAS solutions, the first one is efficient and the second is collusion-resistant.

5.2 Efficient Location Anonymization Scheme Based on Chinese Remainder Theorem

In an election with N voters and two collectors, C_1 generates a prime m_1 of even number of bits such that $m_1 > N$ and C_2 generates a prime m_2 of odd number of bits such that $m_2 > N$. Thus, m_1 and m_2 will be relatively prime. Each collector $\mathcal{C}_j$ creates a set U_j of N random numbers from $\mathbb{Z}_{m_j}$ where $j = 1, 2$. Each voter V_i receives $u_1 \in U_1$ and m_1 from C_1, and $u_2 \in U_2$ and m_2 from C_2 $(u_1, u_2 \leq N)$. According to Chinese Remainder Theorem (CRT) for the congruent system $x = u_1 \mod m_1$ and $x = u_2 \mod m_2$, V_i computes his unique location L_i as follows: $a_i = u_1 m_2 (m_2^{-1} \text{ mod } m_1)$, $b_i = u_2 m_1 (m_1^{-1} \text{ mod } m_2)$, and $L_i = (a_i + b_i) \text{ mod } (m_1 m_2)$.

By the principle of CRT, as long as voters do not get exactly same u_1 and u_2, their L_i will be different. This protocol scales to any number of collectors due to the fact that CRT accommodates a congruence system of more than two equations. As usual, only two collectors are discussed here for the sake of simplicity.

This scheme is clearly efficient. For this scheme to work safely, we assume that both collectors are honest in the sense that none of them would collude with a voter to get another collector's prime. Otherwise, if a collector gets another collector's prime, the collector can figure out which location belongs to which voter. In addition, as long as at least one collector behaves honestly and selects different u_i for different voters, all voters' locations will be different.

Notes: when m_1 and m_2 are large and the number of voters is small (comparatively), the resulted unique locations may be sparse in the range of 0 to $m_1 m_2 - 1$.

The voters run the mutual restraining voting protocol to get a tallied voting vector which has 1 in all chosen locations but 0 in all other locations. Thus, each voter will get a unique consecutive location by counting how many 1s from the beginning until his location (or to say, subtracting from his location the number of zeros ahead of this location). The same applies to the scheme in next section.

5.3 Collusion-resistant Location Anonymization Scheme Based on Tailored STPM

To address the collusion concern in the CRT based LAS described above, we add the collusion-resistant capability to it. Here we assume that there are three collectors. Many elections have more than two candidates, e.g., having an independent candidate besides Republican and Democratic candidates in some US presidential elections. In case there are just two parties, we can easily add one independent auditor as an additional collector. We denote three collectors as C_1, IC, and C_2. The collusion-resistant LAS protocol works as follows.

- IC will generate two relative primes m_1 and m_2 such that $m_1 > N$ and $m_2 > N$. IC will also compute $m_2(m_2^{-1} \bmod m_1)$, $m_1 m_2$, and $m_1(m_1^{-1} \bmod m_2)$.
- When a voter registers, C_1 will select a $u_1 \leq N$ for his and C_2 will select a $u_2 \leq N$ for her.
- C_1 and IC will compute $u_1(m_2(m_2^{-1} \bmod m_1)) = a_1 + a_2$ using the tailored STPM, with C_1 having a_1 and IC having a_2. Similarly, C_2 and IC will compute $u_2(m_1(m_1^{-1} \bmod m_2)) = b_1 + b_2$, with C_2 having b_1 and IC having b_2.
- C_1 will send a_1 to the voter, and C_2 will send b_1 to the voter. IC will send $a_2 + b_2$ and $m_1 m_2$ to the voter.
- The voter then computes $(a_1 + a_2 + b_1 + b_2) \bmod (m_1 m_2)$ to get his unique location. Neither of C_1, IC, and C_2 can find out his location.

5.4 Complexity Analysis and Simulation

Our complexity analysis and simulation focus on the original LAS in Section 5.1. The original LAS uses similar mechanisms of the voting protocol in Chapter 4 during each round. Thus for each round, we obtain similar complexity as analyzed in Section 4.3.1. Roughly, the message length T in LAS is $O(\bar{L})$.

The simulation for the original LAS is conducted on the same platform as in Section 4.3.2.

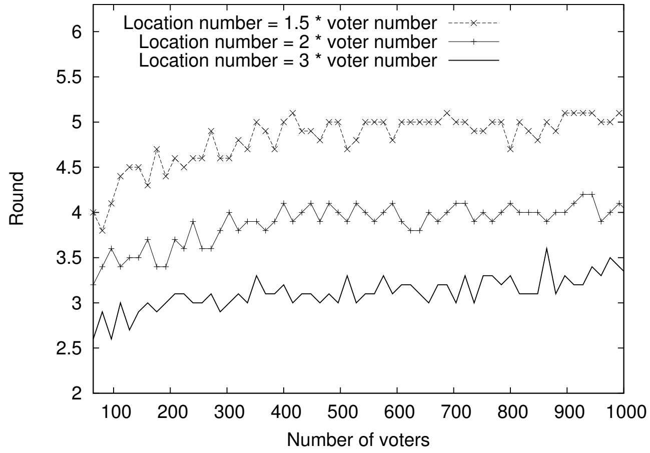


Fig. 5.1. Original LAS: number of rounds needed for location anonymization

Fig. 5.1 shows the number of rounds needed for completing location anonymization. The length of the location vector $\bar{L}$ varied from 1.5, 2, to 3 times of number of voters N. The number of voters N varied from 64 to 1000 by an increment of 16. As shown in Fig. 5.1, the number of rounds needed for completing location anonymization is relatively stable for different N under a given ratio $\bar{L}/N$.

Fig. 5.2 shows the time spent on location anonymization by each voter. The length of the location vector $\bar{L}$ and the number of voters N varied the same as in Fig. 5.1. The length of the location vector is $\bar{L} = O(N)$; the aggregation of location vectors is dominated by the (N, N)-SS. The execution time is $O(N\bar{L}^2)$. Thus, the time spent on location anonymization is $O(N^2)$. When the ratio $\bar{L}/N = 3$, 2 to 3 rounds were

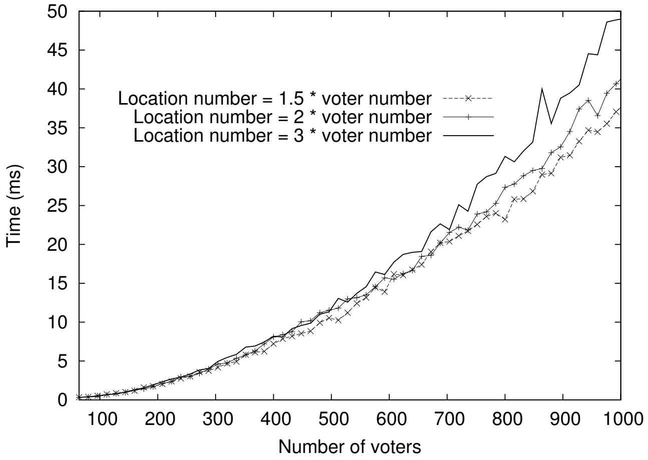


Fig. 5.2. Original LAS: time needed for location anonymization per voter

sufficient for completing location anonymization. For example, with 1000 voters, it took no more than 0.05 seconds to anonymize voters' locations. This demonstrates the efficiency of the proposed LAS.

6. SECURITY AND PROPERTY ANALYSIS

In this chapter, we demonstrate a few important properties and also analyze the robustness of our mutual restraining voting protocol in Chapter 4 and LAS in Chapter 5.

6.1 Analysis of Main Properties

Completeness (Correctness). All votes are counted correctly in this voting protocol. That is, the aggregated voting vector $\mathbf{V_A}$ of the length $L = NA$ in binary is the sum of all individual voting vector $\mathbf{v}_i$ from each voter ($\mathbf{V_A} = \sum_{i=1}^{N} \mathbf{v}_i$). Likewise, $\mathbf{V'_A} = \sum_{i=1}^{N} \mathbf{v}'_i$.

Verifiability. Due to its transparency, the protocol provides a full range of verifiability with four levels:

1. A voter can verify that his secret ballot is submitted correctly.
2. A voter (and any third party) can verify that the aggregated voting vector is computed correctly.
3. A voter can verify that his vote is cast correctly.
4. A voter (and any third party) can verify that the final tally is performed correctly.

About individual verification, different techniques may have different meanings and adopt different mechanisms to implement. For example, majority of typical e-voting techniques encrypt the votes and a voter verifies his cast ballot in an encrypted format [68, 69], rather than in plain text format/clear vote. In this case, tallying is normally done via homomorphic cryptosystem. Due to the fundamental principle of homomorphic encryption, voters should be convinced that the final tally is accurate and their votes are correctly included in the final tally. Some e-voting techniques utilize pairs of (pseudo-voter ID, vote) and a voter verifies his cast vote (in plain format) according to his pseudo-voter ID. The relation between a voter's real identity and his pair is hidden or anonymized via an anonymous channel or mix-net [13, 70]. One representative case of this kind is the technique in [71]. A voter casts his encrypted vote via an anonymous channel, and then sends his encryption key via the same channel for the counter to decrypt/open his vote. The voter can verify his vote in plain format (as well as in encrypted format). In this case, the voter's real identity is hidden by blind signature and anonymous channel. Here the assumption is that the anonymous channel, mix-net and blind signature are trustworthy. Furthermore, for all these verification scenarios, the mechanisms used for anonymization and individual verification act as one kind of black-box and introduce a gap between a voter's ballot and real vote.

Like the technique in [71], our technique allows a voter to verify his vote in plain text format. However, different from [71], the verification in our technique is visibly realized due to full transparency and seamless transition from ballots (no information about any vote) to all individual votes (each clear vote is anonymous to anyone except the vote's owner). No gap exists and no trustworthy assumption is required.

Anonymity. The protocol preserves anonymity if no more than $N-2$ voters collude. This claim follows the proof of Corollary 3.2.3 as p_i is known to all. Also, the protocol splits trust, traditionally vested in a central authority, now between two non-colluding collectors with conflicting interests. One collector does not have enough information to reveal a vote.

Especially in the current Sub-protocol 1, we eliminate the possibility for an attacker to perform brute-force search against the intermediate result as in the original Sub-protocol 1 in [65]. Basically, in the original Sub-protocol 1, two collectors exchange $g^{r_1+r_1'}$ and $g^{r_2+r_2'}$, so both obtain $g^{r_1+r_2'+r_1'+r_2}$ such that

$$\begin{aligned} g^{r_1+r_2'+r_1'+r_2} &= g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}} \\ &= (g^{S_{i,C_2}})^{S'_{i,C_1}} \times (g^{S'_{i,C_2}})^{S_{i,C_1}} \qquad (6.1) \end{aligned}$$

Without loss of generality, let us assume C_1 wants to find out v_i. Since C_1 has S_{i,C_1} and S'_{i,C_1}, $g^{s_{ii}}$ and $g^{s'_{ii}}$ are published, and $v_i \times v'_i = 2^{L-1}$, C_1 guesses $v_i = 2^j$ (with v'_i being $2^{(L-1-j)}$) for $j = 0, 1, \cdots, L-1$, and constructs $g^{\bar{S}_{i,C_2}}$ and $g^{\bar{S}'_{i,C_2}}$ based on Equations (6.2) and (6.3) respectively.

$$g^{S_{i,C_2}} = g^{v_i - s_{ii} - S_{i,C_1}} = g^{v_i}(g^{s_{ii}})^{-1}g^{-S_{i,C_1}} \tag{6.2}$$

$$g^{S'_{i,C_2}} = g^{v'_i - s'_{ii} - S'_{i,C_1}} = g^{v'_i}(g^{s'_{ii}})^{-1}g^{-S'_{i,C_1}} \tag{6.3}$$

C_1 then verifies if $(g^{\bar{S}_{i,C_2}})^{S'_{i,C_1}} \times (g^{\bar{S}'_{i,C_2}})^{S_{i,C_1}}$, corresponding to the right hand side (RHS) of Eqation (6.1), equals to $g^{r_1+r'_2+r'_1+r_2}$, the left hand side (LHS) of Equation (6.1). If they are equivalent, V_i's vote v_i is found to be 2^j. Otherwise, C_1 guesses next $v_i = 2^{j+1}$ until he finds out the correct v_i.

However, in the current Sub-protocol 1 presented here, K_1 contains a random value of $r_1 + r'_1$, and similarly, K_2 has $r_2 + r'_2$. C_1 can compute either $K_2 \times g^{r_1+r'_1}$ as shown in LHS of Equation (6.4), or $K_1 \times K_2$ as shown in LHS of Equation (6.5) to get rid of the randomness. C_1 then guesses v_i as before and plugs it into RHS of Equation (6.4) or Equation (6.5) to verify if the equations hold true. But they will always hold true no matter what value v_i is guessed. Thus, C_1 will not be able to find out v_i with the brute-force search. Vote anonymity is preserved.

$$\begin{aligned}
K_2 \times g^{r_1+r'_1} &= g^{S_{i,C_2}(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})+S'_{i,C_2}(s_{ii}+S_{i,C_1})} = g^{S_{i,C_2}v'_i+S'_{i,C_2}(s_{ii}+S_{i,C_1})} \\
&= g^{(v_i-s_{ii}-S_{i,C_1})v'_i+(v'_i-s'_{ii}-S'_{i,C_1})(s_{ii}+S_{i,C_1})} \\
&= g^{v_i \times v'_i - s_{ii}s'_{ii} - s_{ii}S'_{i,C_1} - s'_{ii}S_{i,C_1} - S_{i,C_1}S'_{i,C_1}} &(6.4) \\
K_1 \times K_2 &= g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})-s_{ii}s'_{ii}} = g^{v_i \times v'_i - s_{ii}s'_{ii}} &(6.5)
\end{aligned}$$

Ballot validity and prevention of multiple voting. The forward and backward mutual lock voting allows a voter to set one and only one of his voting positions to 1 (enforced by Sub-protocol 1).

The ballot of p_i and p'_i is ensured to be generated correctly in the forms of $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$ and $p'_i = s'_{ii} + \tilde{S}'_{i,C_1} + \tilde{S}'_{i,C_2}$ (enforced by Sub-protocol 2).

Fairness. Fairness is ensured due to the unique property of (N, N)-SS: no one can obtain any information before the final tally, and only when all N secret ballots are aggregated, all votes are obtained anonymously. It is this sudden transition that precludes any preannouncement of partial voting results, thus achieving fairness.

Eligibility. Voters have to be authenticated for their identities before obtaining voting locations. Traditional authentication mechanisms can be integrated into the voting protocol.

Auditability. Collectors collaboratively audit the entire voting process. Optionally we can even let collectors publish their commitment to all shares they generate (using hash functions, for example). With the whole voting data together with collectors' commitments, two collectors or a third authority can review the voting process if necessary.

Transparency and voter assurance. Many previous e-voting solutions are not transparent in the sense that although the procedures used in voting are described, voters have to entrust central authorities to perform some of the procedures. Voters cannot verify every step in a procedure [72]. Instead, our voting protocol allows voters to visually check and verify their votes on the bulletin board. The protocol is transparent where voters participate in the whole voting process.

6.2 Robustness Against Voting Misbehavior

The protocol is robust in the sense that a misbehaving voter will be identified. In the interactive voting protocol, a misbehaving voter V_i may:

- submit an invalid voting vector $\mathbf{v}_i$ ($\mathbf{v}'_i$) with more than one (or no) 1s;
- generate wrong s_{ii} (s'_{ii}), thus wrong commitment $g^{s_{ii}}$ ($g^{s'_{ii}}$);
- publish an incorrect secret ballot p_i (p'_i) such that $p_i \neq s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$ ($p'_i \neq s'_{ii} + \tilde{S}'_{i,C_1} + \tilde{S}'_{i,C_2}$).

First, we show that a voter submitting an invalid voting vector $\mathbf{v}_i$ ($\mathbf{v}'_i$) with more than one 1s will be detected. Without loss of generality, we assume two positions, L_c^i and $L_c^{i\prime}$, are set to 1. (A voter can also misbehave by putting 1s at inappropriate positions, i.e., positions assigned to other voters; we will analyze this later.) Thus the voter V_i obtains v_i (v'_i), such that

$$\begin{aligned} v_i &= 2^{(L-L^i_c)} + 2^{(L-L^{i\prime}_c)}, v'_i = 2^{(L^i_c-1)} + 2^{(L^{i\prime}_c-1)}, \\ v_i \times v'_i &= 2^{L-1} + 2^{L-1} + 2^{L-L^i_c+L^{i\prime}_c-1} + 2^{L-L^{i\prime}_c+L^i_c-1}. \end{aligned}$$

All the computations are moduli operations. By using $\mathbf{Z}^*_{\mathbf{A}}$, which has at least $2^{2L} - 2^{L+1} + 1$ elements/bits, we have $v_i \times v'_i \neq 2^{L-1}$, thus $g^{v_i \times v'_i} \neq g^{2^{L-1}}$. Assuming V_i generates an invalid voting vector without being detected, this will lead to the following contradiction by Sub-protocol 1:

$$\begin{aligned} g^{2^{L-1}} &= g^{s_{ii}s'_{ii}} \times (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times (g^{s_{ii}})^{S'_{i,C_2}} \\ &\quad \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}} \\ &= g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})} = g^{v_i v'_i}. \end{aligned}$$

Similar proof applies to an invalid voting vector without 1s.

Next, we show that V_i cannot generate wrong s_{ii} or s'_{ii} such that $s_{ii}+S_{i,C_1}+S_{i,C_2} \neq v_i$ or $s'_{ii} + S'_{i,C_1} + S'_{i,C_2} \neq v'_i$. If Sub-protocol 1 fails to detect this discrepancy, there is: $g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})} = g^{2^{L-1}}$. Since the computation is on $\mathbf{Z}^*_{\mathbf{A}}$, we have: $(s_{ii} + S_{i,C_1} + S_{i,C_2})(s'_{ii} + S'_{i,C_1} + S'_{i,C_2}) = 2^{L-1}$. Given that:

$$\begin{aligned} &s_{ii} + S_{i,C_1} + S_{i,C_2} \neq v_i, \;\; s'_{ii} + S'_{i,C_1} + S'_{i,C_2} \neq v'_i, \\ &(s_{ii} + S_{i,C_1} + S_{i,C_2})(s'_{ii} + S'_{i,C_1} + S'_{i,C_2}) = 2^{L-1}, \end{aligned}$$

there must exist one and only one position $L^{i\prime}_c$ which is set to 1 and $L^{i\prime}_c \neq L^i_c$. This indicates that V_i gives up his own voting positions, but votes at a position assigned to another voter V_j ($i \neq j$). In this case, V_i's voting positions in $\mathbf{V_A}$ and $\mathbf{V'_A}$ will be 0[1]. This leads to an invalid tallied vector where V_i's voting positions have all 0s and possibly V_j's have multiple 1s. If this happens, C_1 and C_2 can collaboratively find V_i's row that has all 0s in the voting vector (arranged in an $N \times M$ array).

---

[1] Unless, of course, another voter puts a 1 in V_i's position. We can either trace this back to a voter that has all 0s in his positions, or there is a loop in this misbehaving chain, which causes no harm to non-misbehaving voters.

Third, we show that a voter cannot publish an incorrect p_i (p'_i) to disturb the tally. Given that a misbehaving V_i publishes p_i (p'_i) such that $s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2} \neq p_i$ ($s'_{ii} + \tilde{S}'_{i,C_1} + \tilde{S}'_{i,C_2} \neq p'_i$), we obtain $g^{s_{ii}+\tilde{S}_{i,C_1}+\tilde{S}_{i,C_2}} \neq g^{p_i}$ ($g^{s'_{ii}+\tilde{S}'_{i,C_1}+\tilde{S}'_{i,C_2}} \neq g^{p'_i}$) which will fail in Sub-protocol 2. Note that $g^{s_{ii}}$ and $g^{s'_{ii}}$ have passed the verification of Sub-protocol 1, and $\tilde{S}_{i,C_1}$ and $\tilde{S}_{i,C_2}$ (also, $\tilde{S}'_{i,C_1}$ and $\tilde{S}'_{i,C_2}$) are computed by two collectors with conflict of interests. Thus, there is no way for the voter to publish an incorrect p_i (p'_i) without being detected.

The discussion shows all these misbehaviors should be caught by the collectors using Sub-protocol 1 or Sub-protocol 2. However, assume two cases as below:

- One misbehavior mentioned above mistakenly passes both Sub-protocol 1 and Sub-protocol 2;
- V_i does give up his own voting locations and cast vote at V_j's locations ($i \neq j$).

For Case 1, the possibility leading to a valid voting vector is negligibly small as we will discuss in Section 6.5. Even if the voting vector is valid, any voter can still verify whether the vote in his own location is correct. This is the individual verifiability our protocol provides.

For Case 2, if V_i casts the same as V_j at V_j's locations, there will be a carry, ending up one 1, but not the vote V_j has cast. If V_i casts a vote different from V_j, there will be two 1s at V_j's locations. Because all V_i's locations now have 0s, the tallied voting vector will be invalid for both scenarios. Furthermore, V_j can detect it since his vote has been changed. Again, as we assumed earlier, there is no reason/incentive for a voter to give up his own voting right and disturb other unknown voters.

The analysis above also applies to the non-interactive voting protocol.

6.3 Robustness Against Collusion

Here we analyze the robustness against different collusions and attacks. With the assumption that collectors have conflicting interests, they will not collude, so we exclude such a scenario.

6.3.1 Robustness Against Collusion Among Voters

In the interactive protocol, p_i and $g^{s_{ii}}$ are known. The colluding voters may try to find unknown shares, as illustrated in Table 4.1 in column-wise, and then use them to find out a voter's vote. Given p_i and $g^{s_{ii}}$, based on Corollary 3.2.3, k equals to 1 in this case, so the protocol is robust against collusion among no more than $N-3$ voters. Even if one share is known, and is used with $g^{s_{ii}}$ to infer v_i (illustrated in Table 4.1 in row-wise). Based on Corollary 3.2.2, $N-3$ voters still cannot find out v_i. So the interactive protocol is robust as long as no more than $N-3$ voters (passive adversaries) collude.

In the non-interactive protocol, a voter V_i receive one sum, S_{i,C_j}, of a partial set of shares (i.e., $(N-1)/2$ shares in a row in Table 4.1) from each collector C_j to calculate s_{ii}, and another sum, $\tilde{S}_{i,C_j}$, of a partial set of shares (i.e., $(N-1)/2$ shares in a column in Table 4.1) to get p_i. Having total of $4N$ sums for N voters, they can build $4N$ equations to solve N^2-N variables in a matrix (excluding shares s_{ii} in the main diagonal). However, even N voters together cannot find out all N^2-N shares when $N>5$. After the tally is open, collusion of $N-1$ voters can find out the vote of non-colluding voter. However, the non-interactive protocol is robust as long as no more than $N-2$ voters (passive adversaries) collude.

The protocol is robust against cheating by colluding voters such as double or multiple voting (active adversaries). Colluding voters want to disrupt the voting process. However, even they collude, the commitments and the secret ballots of each colluding voter have to pass the verification of both Sub-protocol 1 and Sub-protocol 2 by two collectors. Thus the disruption will not succeed as discussed in Section 6.2.

6.3.2 Robustness Against Collusion Among Voters and a Collector

Without loss of generality, let's assume that C_1 is the colluding collector.

In the interactive protocol, C_1 has $N/2$ shares in some rows and columns as illustrated in Table 4.1. Correspondingly, C_2 has $N/2-1$ shares in these rows and columns. C_1 has $g^{\tilde{S}_{i,C_2}}$ from C_2. Since $\tilde{S}_{i,C_2}$ is the sum of $N/2-1$ shares, based on Corollary 3.2.3, $k=0$ here, collusion with $N/2-1-2=N/2-3$ will not disclose any information of the unknown shares. But collusion with $N/2-2$ voters will find the remaining one unknown share. In this case, C_1, having $N/2$ shares, together with the colluding $N/2-2$ voters, can use this share and $g^{s_{ii}}$ to find out V_i's vote. However as said earlier, collusion among a collector and no more than $N/2-3$ voters will not be able to disclose any vote information.

In the non-interactive protocol, as discussed in the previous section, for N voters receiving $2N$ sums of shares from the non-colluding collector C_2, it is not possible to find out the individual shares generated by C_2 when $N>5$. Even a collusion of $N-2$ voters still cannot find out a voter's vote because the shares of two non-colluding voters are unknown and cannot be deduced from $\tilde{S}_{i,C_2}$ or any other sources. Thus, a collusion of a collector with no more than $N-2$ voters will not disclose any vote information.

For both interactive and non-interactive protocols, when voters collude with a collector to disrupt the voting by cheating, each individual voter still has to pass Sub-protocol 1 and Sub-protocol 2 by two collectors. However, since one collector is colluding, the voter may succeed in passing the verification.

Assume V_i colludes with C_1. V_i generates $\bar{s}_{ii}$ and $\bar{s}'_{ii}$ (deviating from authentic s_{ii} and s'_{ii}) and publishes commitments $g^{\bar{s}_{ii}}$, $g^{\bar{s}'_{ii}}$, and $g^{\bar{s}_{ii}\bar{s}'_{ii}}$. For Sub-protocol 1, C_1 can derive $\bar{S}_{i,C_1}$ and $\bar{S}'_{i,C_1}$ (deviating from authentic S_{i,C_1} and S'_{i,C_1}) based on V_i's $\bar{s}_{ii}$ and $\bar{s}'_{ii}$, such that:

$$(\bar{s}_{ii}+\bar{S}_{i,C_1}+S_{i,C_2})(\bar{s}'_{ii}+\bar{S}'_{i,C_1}+S'_{i,C_2})=2^{L-1}$$

Similarly, V_i's $\bar{p}_i$ and $\bar{p}'_i$ (deviating from authentic p_i and p'_i) can also pass Sub-protocol 2 by colluding with C_1.

However, since there are non-colluding voters, the probability of leading to a valid tallied voting vector is negligibly slim as we will discuss in Section 6.5. Even by any chance a valid tallied voting vector is created, any voter can still tell if the vote in the final vector is what he intended by the property of individual verifiability.

As a result, such collusion with the purpose of cheating will be detected too.

6.4 Robustness Against Outside Attack

The protocol is robust against an outsider inferring any vote information. The outsider does not have any information. If he colludes with insiders (voters), he will not gain any information as long as no more than $N-2$ voters collude with him.

The protocol is also robust against an outsider disrupting the voting. First, if the outsider intercepts data for a voter from secure channel during the voting, he will not learn any information about vote because the data itself is encrypted. Second, if the outsider wants to change a voter's vote or even the tallied voting vector by colluding with voters or a collector, he will not succeed as discussed in Section 6.3.

6.5 Security Analysis of Attacks Affecting Tallied Voting Vector Without Detection

Theorem 6.5.1 Error detectability: *The probability that a random attack at any point of the voting process in this protocol can succeed without being detected (i.e., the tallied voting vector remains valid) is* $M^N/2^{MN} = 1/2^{(M-logM)N}$.

Proof Suppose an attacker wants to attack the voting system and consequently the tallied voting vector without being detected. In the mutual restraining voting system, an attack possibly passing the detection is the one that results in a valid tallied voting vector. For N voters and M candidates with the voting vector bit length of $N \times M$, a valid tallied voting vector $\mathbf{V_A}$ should have only one 1 in any of N locations, i.e., only one 1 in M bits. Thus, the number of total valid tallied vectors is $M \times M \times \cdots \times M = M^N$. Meanwhile, the number of total possible tallied vectors is 2^{MN}, with each of MN bits being either 0 or 1. It is reasonable to say that an attack which is launched by an attacker at any point of the voting process, no matter how powerful the attacker is, will result in change of some ballots and votes, thus, affecting the final tallied voting vector. We also assume that an attacker can only manipulate partial ballots, not all ballots, and an attacker cannot have exact control on the aggregation of all ballots that generates a valid tallied vector $\mathbf{V_A}$. The attacker may be lucky that a valid tallied vector $\mathbf{V_A}$ is generated. However, the probability that a random tallied voting vector $\mathbf{V_A}$ being valid is $M^N/2^{MN} = 1/2^{(M-logM)N}$. ■

Since $M \geq 2$ and N is large, the probability of any attack without being detected is negligibly small. As an example, with $M = 2$ and $N = 1000$, the probability is 2^{-1000}!

Furthermore, even if a valid tallied voting vector $\mathbf{V_A}$ is generated, there must be certain location containing a vote which does not match what the voter in this location has voted for. Thus, it can be detected through individual verification and reported by the voter who owns this location.

6.6 Correctness and Security Analysis of Two Sub-protocols

We now analyze two sub-protocols used in the voting process.

6.6.1 Correctness and Security Analysis of Sub-protocol 1

Theorem 6.6.1 Correctness of v_i and v'_i: ***Sub-protocol 1*** *allows collectors to check cooperatively the correctness of* v_i *and* v'_i *such that* $K_1 \times K_2 \times g^{s_{ii}s'_{ii}} = g^{v_i v'_i} = g^{2^{L-1}}$.

Proof We show that the collectors are able to jointly verify whether V_i generates correct s_{ii} and s'_{ii}.

As given in Step (d) of **Sub-protocol 1**, C_1 and C_2 exchange K_1 and K_2, so each calculates $K_1 \times K_2$ as follows.

$$\begin{aligned}
K_1 \times K_2 &= (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times g^{r_1+r'_1} \\
&\quad \times (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_2+r'_2} \\
&= (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times g^{r_1+r'_2} \\
&\quad \times (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r'_1+r_2} \\
&= (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_2}} \\
&\quad \times (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}} \\
&= g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})-s_{ii}s'_{ii}} \qquad (6.6)
\end{aligned}$$

Since $g^{s_{ii}s'_{ii}}$ is published by V_i, a collector multiplies this term on both sides of the above equation (6.6), and gets:

$$\begin{aligned}
K_1 \times K_2 \times g^{s_{ii}s'_{ii}} &= g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})-s_{ii}s'_{ii}} \times g^{s_{ii}s'_{ii}} \\
&= g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})}
\end{aligned}$$

If s_{ii} and s'_{ii} are generated correctly by V_i, we should have:

$$v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}, \quad \text{and} \quad v'_i = s'_{ii} + S'_{i,C_1} + S'_{i,C_2}$$

Thus, the following equation should hold true, since $v_i \times v'_i = 2^{L-1}$:

$$g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s'_{ii}+S'_{i,C_1}+S'_{i,C_2})} = g^{v_i \times v'_i} = g^{2^{L-1}} \qquad (6.7)$$

In other words, if V_i does not generate the shares s_{ii} and s'_{ii} correctly, Equation (6.7) will fail the verification conducted by collectors. ■

Theorem 6.6.2 Privacy: *In **Sub-protocol 1**, a collector, given additional information obtained from the other collector through exchange, will not be able to find a voter's vote.*

Proof We prove that collectors will not be able to find V_i's vote v_i from the published information as well as the exchanged information K_1 and K_2.

For a collector, in order to obtain V_i's $v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}$, there are three possible ways to approach: (1) by somehow obtaining s_{ii}, S_{i,C_1}, and S_{i,C_2} and then summing them. Clearly, from all available information and due to the DLP problem, a collector, say C_1, cannot get s_{ii} and cannot get S_{i,C_2} either. Thus, a collector cannot obtain v_i. (2) by guessing a vote v_i and checking whether v_i, s_{ii}, S_{i,C_1}, and S_{i,C_2} satisfy the equation $v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}$. Guessing v_i can certainly be done by trying every possible vote 2^j for $j = 0, \cdots, NM - 1$ where NM is the length of the voting vector. However, the verification of v_i cannot be done through the equation directly since three values, s_{ii}, S_{i,C_1}, and S_{i,C_2}, cannot be obtained all together. Instead, it can only be done through the form below:

$$g^{2^j} = g^{s_{ii}} g^{S_{i,C_1}} g^{S_{i,C_2}} \tag{6.8}$$

A collector, say C_1, has $g^{s_{ii}}$ and $g^{S_{i,C_1}}$ but cannot get $g^{S_{i,C_2}}$ from the additional information K_2 exchanged from C_2. What C_1 can do is as follows: for each $v_i = 2^j$, computes $\bar{S}_{i,C_2} = 2^j - s_{ii} - S_{i,C_1}$, plugs into the above equation, but gets $g^{2^j} = g^{2^j}$. This will always be true no matter what j is. And (3) by considering both $v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}$ and $v'_i = s'_{ii} + S'_{i,C_1} + S'_{i,C_2}$ together, guessing every possible vote, and checking whether the following equation holds. From the exchanged information $K_2 = (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_2+r'_2}$ which contains four terms, C_1 cannot get any single term, or any partial product of three or fewer terms. Moreover, $r_2 + r'_2$ (thus $g^{r_2+r'_2}$) itself is a random value unrelated to the vote. It is necessary for C_1 to get ride of this randomness which can only be finished by multiplying $g^{r_1}g^{r'_1}$. That is:

$$\begin{aligned} K_2 \times g^{r_1} \times g^{r'_1} &= (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_2+r'_2} \times g^{r_1} \times g^{r'_1} \\ &= (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_1+r'_2} \times g^{r'_1+r_2} \\ &= (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}} \end{aligned} \tag{6.9}$$

We call the five terms in Equation (6.9), $(g^{s_{ii}})^{S'_{i,C_2}}$, $(g^{s'_{ii}})^{S_{i,C_2}}$, $g^{S_{i,C_2}S'_{i,C_2}}$, $g^{S_{i,C_1}S'_{i,C_2}}$, and $g^{S'_{i,C_1}S_{i,C_2}}$, the *minimum set of terms* in order to do a brute-force search. In other words, C_1 cannot single out any individual term, or product of any two or three or four terms, from Equation (6.9).

Now C_1 guesses v_i to be 2^j and v'_i to be 2^{L-j-1} for $j = 0, \cdots, L-1$ where L is the length of the voting vector, and constructs $\bar{S}_{i,C_2}$ and $\bar{S}'_{i,C_2}$ as follows to replace S_{i,C_2} and S'_{i,C_2} in Equation (6.9), respectively.

$$\bar{S}_{i,C_2} = 2^j - s_{ii} - S_{i,C_1}, \quad \text{and} \quad \bar{S}'_{i,C_2} = 2^{L-j-1} - s'_{ii} - S'_{i,C_1}$$

After substituting S_{i,C_2} and S'_{i,C_2} in Equation (6.9) with $\bar{S}_{i,C_2}$ and $\bar{S}'_{i,C_2}$, we have:

$$\begin{aligned} K_2 \times g^{r_1} \times g^{r'_1} &= (g^{s_{ii}})^{2^{L-j-1}-s'_{ii}-S'_{i,C_1}} \times (g^{s'_{ii}})^{2^j-s_{ii}-S_{i,C_1}} \\ &\quad \times g^{(2^j-s_{ii}-S_{i,C_1})(2^{L-j-1}-s'_{ii}-S'_{i,C_1})} \\ &\quad \times g^{S_{i,C_1}(2^{L-j-1}-s'_{ii}-S'_{i,C_1})} \times g^{S'_{i,C_1}(2^j-s_{ii}-S_{i,C_1})} \\ &= g^{2^j \times 2^{L-j-1}-s_{ii}s'_{ii}-s_{ii}S'_{i,C_1}-s'_{ii}S_{i,C_1}-S_{i,C_1}S'_{i,C_1}} \\ &= g^{2^{L-1}} \times (g^{s_{ii}s'_{ii}})^{-1} \times (g^{s_{ii}})^{-S'_{i,C_1}} \\ &\quad \times (g^{s'_{ii}})^{-S_{i,C_1}} \times g^{-S_{i,C_1}S'_{i,C_1}} \end{aligned} \tag{6.10}$$

From Equation (6.10), we find that all terms having j have already been canceled out. In other words, no matter what j is, Equation (6.10) always holds true. So we conclude that C_1 cannot guess V_i's vote v_i from all the given information. ■

6.6.2 Correctness and Security Analysis of Sub-protocol 2

Theorem 6.6.3 Correctness of p_i and p'_i: ***Sub-protocol 2*** *allows any collector to check the correctness of* p_i *such that* $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$, *and similarly for* p'_i *such that* $p'_i = s'_{ii} + \tilde{S}'_{i,C_1} + \tilde{S}'_{i,C_1}$.

Proof As given in Section 4.2, V_i publishes $g^{s_{ii}}$. After C_1 and C_2 compute and exchange $g^{\tilde{S}_{i,C_1}}$ and $g^{\tilde{S}_{i,C_1}}$, any of them can compute and verify that:

$$g^{p_i} = g^{s_{ii}} \times g^{\tilde{S}_{i,C_1}} \times g^{\tilde{S}_{i,C_2}} = g^{s_{ii}+\tilde{S}_{i,C_1}+\tilde{S}_{i,C_2}}$$

Similarly, any collector can also compute and verify that:

$$g^{p'_i} = g^{s'_{ii}} \times g^{\tilde{S}'_{i,C_1}} \times g^{\tilde{S}'_{i,C_2}} = g^{s'_{ii}+\tilde{S}'_{i,C_1}+\tilde{S}'_{i,C_2}}$$

■

Theorem 6.6.4 Privacy: *In **Sub-protocol 2**, a collector, given additional information published by the other collector, will not be able to find information about vote.*

Proof We prove that any collector such as C_1 will not be able to find V_i's vote v_i with additional information exchanged from C_2. Since $v_i = s_{ii} + S_{i,C_1} + S_{i,C_2}$, in order to get v_i, C_1 needs to get s_{ii} and S_{i,C_2} first. (1) Due to the DLP property, C_1 cannot get s_{ii} from the published commitment $g^{s_{ii}}$. (2) From $g^{\tilde{S}_{i,C_2}}$ given by C_2, C_1 cannot get any information of s_{ii} and S_{i,C_2} since $\tilde{S}_{i,C_2}$ contains shares of other voters' votes. (3) Although C_1 has $\tilde{S}_{i,C_1}$ and p_i where $p_i = s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$, since C_1 cannot get $\tilde{S}_{i,C_2}$ from $g^{\tilde{S}_{i,C_2}}$, C_1 cannot get s_{ii} from $\tilde{S}_{i,C_1}$ and p_i either. As a result, any collector cannot get V_i's vote v_i from the published and exchanged information.

■

6.7 Analysis of Location Anonymization Schemes

In this section, we analyze the properties of the location anonymization schemes in Chapter 5.

6.7.1 Robustness of Original Location Anonymization Scheme

The analysis in Section 6.2 shows that no voter can choose more than one positions during the location anonymization process. However, this does not address the problem that a malicious participant deliberately induces collisions by choosing a location that is already occupied by another voter. We will demonstrate that our proposed LAS is robust against this.

Let the collision happen at $\hat{L_i}$, i.e., $\hat{L_i}$ is chosen by V_i in the previous round, and both V_i and V_j claim $\hat{L_i}$ in the current round. In this case, V_j is the voter who deliberately introduces collision. To identify a voter who chooses $\hat{L_i}$ in a given round, C_1 and C_2 do the following collaboratively. For each voter, using the tailored STPM, C_1 and C_2 compute $Q = g^{g^{S_{i,C_1}+S_{i,C_2}}}$ ($Q' = g^{g^{S'_{i,C_1}+S'_{i,C_2}}}$) and check if $g^{g^{10^{\tilde{\hat{L}}-\hat{L_i}}}/g^{s_{ii}}} = Q$ ($g^{g^{10^{\hat{L_i}-1}}/g^{s'_{ii}}} = Q'$). By doing this, the collectors identify the voter who selects $\hat{L_i}$ without divulging others' locations. Although the honest voter V_i who chooses $\hat{L_i}$ is exposed along with the malicious V_j, V_i can restore location anonymity by selecting another location in the next round and V_j should be punished.

Of course, voters may collude to infer location information. If k voters collude, they will know that the rest non-colluding $N-k$ voters occupy the remaining $N-k$ voting locations. Since we assumed in Section 4.1.1 that majority of voters is benign, we consider the leaking of location information in this case is acceptable and will not endanger the voting process.

6.7.2 Correctness and Security Analysis of Chinese Remainder Theorem Based Location Anonymization Schemes

Theorem 6.7.1 Correctness: *Two CRT based LAS each compute unique non-colliding locations for voters correctly.*

Proof Each location is the solution to the system of two congruent equations $L_i = u_1 \bmod m_1$ and $L_i = u_2 \bmod m_2$. Different pairs of (u_1, u_2) will have different L_i's, which is guaranteed by the CRT principle. Thus, as long as at least one collector is honest and selects a different u_i for different voters, L_i will be unique. ■

Theorem 6.7.2 Privacy of voter's locations: *The CRT based collusion-resistant LAS is able to generate a secret location for a voter which no one but the voter knows.*

Proof First, let us consider any collector, say, C_1. Due to the tailored STPM, C_1 knows u_1 and gets a_1, but cannot get any information of u_2, m_1, or m_2. So, any collector will not be able to get a voter's location. Second, let us consider IC. IC knows m_1, m_2, a_2, and b_2, but he cannot get any information of u_1 or u_2. Therefore, IC cannot get a voter's location either. Third, for any voter, he knows m_1m_2, so he can compute his own location from $a_1 + b_1 + a_2 + b_2 \bmod m_1m_2$. But he does not know u_1 or u_2 of any voters (even his own u_1 and u_2), so he cannot know the locations of any other voters. ■

Theorem 6.7.3 Collusion-resistance: *The CRT based collusion-resistant LAS is able to defend against the collusion of a collector and a voter.*

Proof First, let us consider the case that any collector, say, C_1, colludes with a voter. C_1 will know u_1, a_1, b_1, a_2+b_2, m_1m_2. Recall that $a_1+a_2 = u_1(m_2(m_2^{-1} \bmod m_1)$, $b_1+b_2 = u_2(m_1(m_1^{-1} \bmod m_2)$. Logically, this information corresponds to two equations involving three variables, so C_1 (and the voter) cannot get u_2, m_1, or m_2 directly by trying to solve the equations. Another way is that given m_1m_2, C_1 tries to factor it to get m_1 and m_2. Assume that m_1 and m_2 chosen by IC are large enough, so C_1 will not be able to do factorization due to its difficulty.

Second, let us consider the case that IC colludes with a voter. IC has m_1 and m_2, and the voter tells IC his computed location L_i. From L_i, IC can get $u_1 = L_i \bmod m_1$ and $u_2 = L_i \bmod m_2$. Except these, IC (and the voter) has no information about u_1 and u_2 selected by C_1 and C_2 for other voters. So such a collusion is meaningless. ■

7. ENABLING RECEIPT-FREENESS AND COERCION-RESISTANCE

In the mutual restraining e-voting model, since the tallied voting vector $\mathbf{V_A}$ contains all individual votes, a voter's secret location can become a self-claimed receipt exploited potentially for vote-selling. It is also possible for a coercer to coerce a voter to disclose his secret location. We argued that after all votes are open, the coerced voter can give the coercer a random location having the requested vote. Here we address a more advance case where the coercer requests the secret location before the final tally. In thise case, if the voter gives a random location other than his own, the vote at this random secret location may happen to be what the coercer wants to cast (with the probability of $1/M$). However, there can be as much as $(M-1)/M$ probability that the vote is different, assuming that each candidate is equally voted. In this chapter, we propose solutions to these vote-selling and voter-coercion problems.

7.1 Solutions to Vote-selling and Individual Verification

We first address the vote-selling issue. Instead of publishing p_i from each voter and $\mathbf{V_A}$ as described in Chapter 4, V_i sends p_i to collectors only. In addition, the collectors *randomly shift* or *permute* $\mathbf{V_A}$ first, and then publish the skewed voting vector, denoted as $\mathbf{SV_A}$. The mechanism will prevent a voter from using his secret location as a receipt, thus resolving the vote-selling problem.

Here we assume that collectors are benign and they randomly pick an integer k where $1 < k < N$ and compute the skewed voting vector $\mathbf{SV_A}$ from $\mathbf{V_A}$ as follows.

$$\begin{aligned}\mathbf{SV_A} &= \mathbf{V_A}/2^{kM} + \mathbf{V_A} \times 2^{L-kM} \mod 2^L \\ &= (\mathbf{V_A} + \mathbf{V_A} \times 2^L)/2^{kM} \mod 2^L\end{aligned} \tag{7.1}$$

where the first term in Equation (7.1) is right shifted by kM bits and the second term is left shifted by $L-kM$ bits. kM is used so that a voter's vote will not be separated off in the middle. Intuitively, $\mathbf{V_A}$ is now divided into a left part $\mathbf{V_{A_l}}$ and a right part $\mathbf{V_{A_r}}$ at the kMth bit such that $\mathbf{V_A} = \mathbf{V_{A_l}}||\mathbf{V_{A_r}}$ where $||$ is the concatenation operation. $\mathbf{SV_A}$ is then the flip of $\mathbf{V_{A_l}}$ and $\mathbf{V_{A_r}}$, i.e., $\mathbf{SV_A} = \mathbf{V_{A_r}}||\mathbf{V_{A_l}}$.

Clearly, like most receipt-free and coercion-resistant protocols such as [70], implementing both receipt-freeness and individual verification normally results in a conflict. The skewed voting vector $\mathbf{SV_A}$ will also nullify the ability for honest voters to verify their votes on the bulletin board. To address this individual verification issue, we propose the following mechanism based on *1-out-of-N oblivious transfer* ($OT(1,N)$).

1. The collector (one collector or both collectors together) will translate the tallied voting vector into N votes, i.e., $u_1, \cdots, u_i, \cdots, u_N$ where $1 \leq u_i \leq M$ and $1 \leq i \leq N$.

2. Suppose a voter has his secret location L_i where $1 \leq L_i \leq N$ and the voter wants to verify his vote (i.e., u_{L_i}). The voter and a collector will use $OT(1,N)$ to perform individual verification. Optionally, we can let a voter conduct verification using $OT(1,N)$ with each collector independently to check if the returned values are equal.

The detail is as follows (suppose that the collector creates a RSA public key e and a private key d, modulus $\hat{n}$):

1. A collector selects N random values $r_1, \cdots, r_N$ and sends all to the voter.

2. The voter V_i picks r_{L_i} corresponding to his location L_i, selects a random value k, blinds and computes $r = (r_{L_i} + k^e) \bmod \hat{n}$ and sends r to the collector.

3. The collector, not knowing which r_i should be used for unblinding k^e, computes $k_1 = (r - r_1)^d, \cdots, k_i = (r - r_i)^d, \cdots,\ k_N = (r - r_N)^d$. Note that k_{L_i} will be k (but the collector does not know) and all others will be random values that does not reveal any information about k.

4. The collector computes and sends $c_1 = v_1 + k_1, \cdots, c_i = v_i + k_i, \cdots, c_N = v_N + k_N$ to the voter.

5. The voter, knowing k_{L_i} must be k, gets $v_{L_i} = c_{L_i} - k$.

Notes: (1) Individual verification (IV) requires voter authentication first. (2) IV is an interactive process between the voter and the collector. In addition, we can let the voter conduct IV using $OT(1, N)$ with each collector, and two IVs can be performed independently. Both collectors must behave honestly. Otherwise, the two independent IVs may not match with the probability of $(N - 1)/N$. (3) For each voter, we assume that he can only do one IV with one of the collectors (or two IVs each with one collector). This will prevent a voter from selecting and decrypting a vote other than his own. Of course, there is no incentive for a voter to get another vote (not his own) since the voter of that vote is anonymous to her. For avoiding the voter to be coerced by an attacker during IV, special physical place/room can be set so voters enter it to verify their votes by the above $OT(1, N)$ IV protocol privately.

Some researchers argued that individual verification is not imperative for every voter. Instead, it is only necessary that a random sample of voters verify their votes to ensure that the system behaves correctly [72].

Other possible IV solutions (typically called probabilistic or randomized individual verification [72]) are:

- collectors extract some random locations (by the protocol above) and display their locations and corresponding votes. Voters (who own these locations but anonymous to collectors) can verify their votes.

- Voters are divided into voting groups, with each of the groups executing the voting protocol. Collectors randomly select some voting groups to disclose their votes. So voters in these groups can verify their votes.

Through the randomized vote verification, all voters or groups can be convinced with great confidence that the system works as expected.

7.2 Solution to Voter-coercion

We utilize the *fake credential* concept in [13] to combat voter-coercion. Let each voter have two (or multiple) secret locations in the voting vector: one location (i.e., real location) for real vote and the other(s) (i.e., fake location(s)) for fake vote(s). To defeat the Over-the-Shoulder Coercer [70] in this e-voting system, a coerced voter can simply cast the coerced vote at a fake location (like a panic password in [70]) other than his real location during the presence of a coercer, and then later he casts his real vote at his real location when the coercer is absent (as assumed in [70], remote e-voting allows a voter to cast his vote anywhere). If multiple votes are cast by a voter, the collectors can simply tally the real vote and ignore the other fake vote(s).

7.3 Individual Verification With Receipt-freeness

There is one noticeable difference in individual verification. In the protocol presented in Chapter 4, every voter can visually verify his vote in the tallied voting vector $\mathbf{V_A}$ published on the bulletin board. In the solution to vote-selling described in Section 7.1, the tallied voting vector $\mathbf{V_A}$ has been skewed, so individual verification is implemented differently, such as using $OT(1, N)$ and/or random sampling. These individual verification techniques substitute the visual verification on the bulletin board. They prevent a voter from using the location as a receipt to prove the vote on the bulletin board to a third party in the protocol described in Chapter 4.

8. PROTOTYPE IMPLEMENTATION

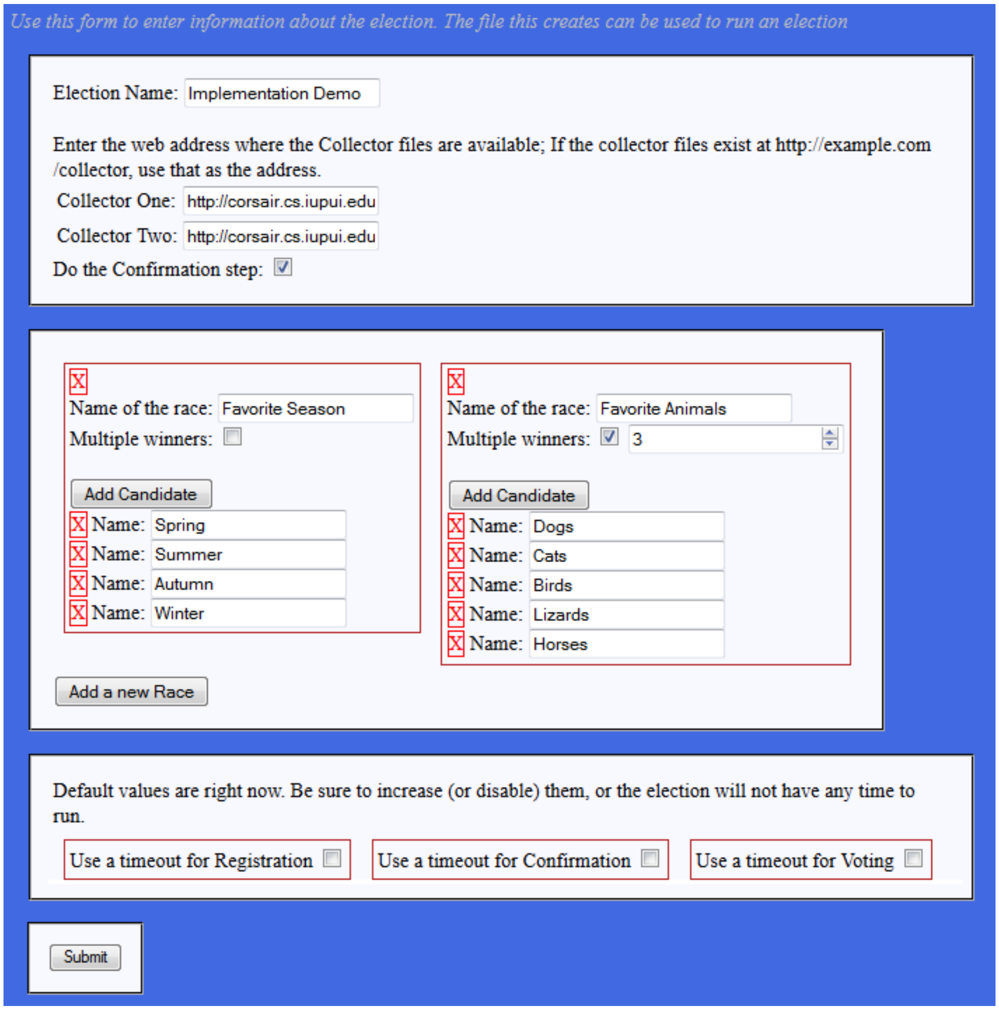
We developed a prototype application suite [73] that allows for functional demonstration/use of the non-interactive protocol presented in Section 4.2.3. Implemented in PHP and Python, it runs on any web server equipped with those interpreters. While it clearly demonstrates the principles and adheres to the cryptographic protocols described earlier, it is still a prototype, and is in active development.

The application allows both single winner and multiple winners, and they can be set up in one election, as shown in the example here. In the case of multiple winners, multiple distinct choices come from a pool of candidates, and are ranked by their respective total votes.

8.1 Roles

Three distinct groups of people are involved in this prototype application. The **administrator** defines the election itself: what races are being voted on, who the candidates are, whether or not a given race is one of the multi-winner cases, what the time limits are (if there are any) to each stage of the election, and who the collectors are (in the form of web addresses). This information is stored in the JSON format and distributed to the collectors using a web application pictured in Fig. 8.1.

The **collectors** fulfill the role described in Chapter 4: they maintain a database that keeps track of voter usernames, the keys they use to encrypt their ballots, and the uploads they submit. With the exception of the keys, the information is publicly viewable as discussed in Section 8.3. They run an application that generates their part of the share matrix and the numbers used in location finding, calculates each voter's partial key, tallies the location vector if location consolidation is in use, and tallies the actual election's results. They also maintain a series of web applications



Use this form to enter information about the election. The file this creates can be used to run an election

Election Name: Implementation Demo

Enter the web address where the Collector files are available; If the collector files exist at http://example.com /collector, use that as the address.

Collector One: http://corsair.cs.iupui.edu

Collector Two: http://corsair.cs.iupui.edu

Do the Confirmation step: ☑

Name of the race: Favorite Season

Multiple winners: ☐

☒ Name: Spring

☒ Name: Summer

☒ Name: Autumn

☒ Name: Winter

Name of the race: Favorite Animals

Multiple winners: ☑ 3

☒ Name: Dogs

☒ Name: Cats

☒ Name: Birds

☒ Name: Lizards

☒ Name: Horses

Default values are right now. Be sure to increase (or disable) them, or the election will not have any time to run.

Use a timeout for Registration ☐

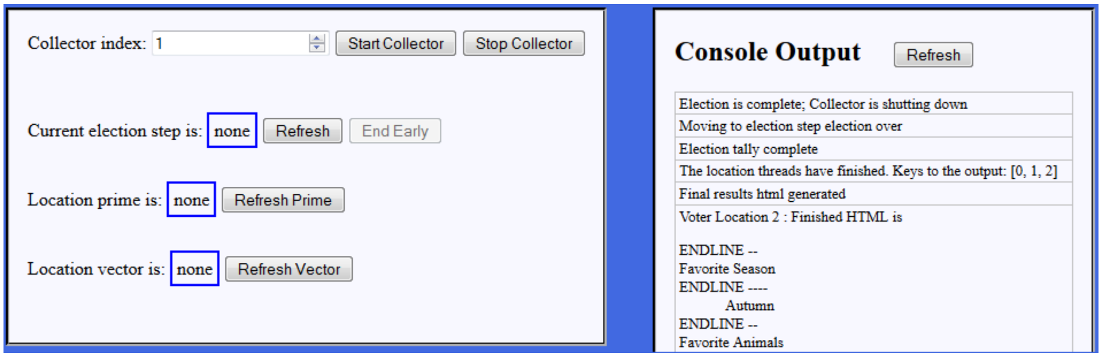
Use a timeout for Confirmation ☐

Use a timeout for Voting ☐

Fig. 8.1. Administrator uses the election generator to create and distribute the election file

for the voters and anyone monitoring the election. All of these tasks are handled automatically, though a web page exists for monitor and control. This control panel is pictured in Fig. 8.2.

The **voters** are the people who actually vote for the candidates. They are identified by a unique username, which may be made public, is secured with a password in this prototype application. At each stage of the election, the voters use a web page to give the needed information to the collectors. As the collectors operate separately



Collector index: 1

Current election step is: none

Location prime is: none

Location vector is: none

```
Election is complete; Collector is shutting down
Moving to election step election over
Election tally complete
The location threads have finished. Keys to the output: [0, 1, 2]
Final results html generated
Voter Location 2 : Finished HTML is

ENDLINE --
Favorite Season
ENDLINE ----
          Autumn
ENDLINE --
Favorite Animals
```

Fig. 8.2. Collector control panel

from each other, the page the voters use packages the voter's data securely and sends it to each of the collectors via asynchronous javascript.

8.2 Stages

Activity is broken into four stages, three of which involve the voters, one of which is optional, with a strictly observational fifth stage at the end.

Preliminary: Before the election is opened to voters, the administrator must create the election file as described earlier and distribute it to the collectors. The collectors, meanwhile, must set up all necessary files on their servers. A web application has been developed that allows the administrator to easily specify necessary information for the election file, and which automatically distributes that file to the collectors. Once obtained, the collectors initialize their databases and open registration.

Registration: This is the first stage of the election that the voters directly participate in. They give a username and password and send those to the collectors. If the username is available, it is stored in the databases, and that voter is now registered. For private elections, this step could be integrated with a security check or could be bypassed altogether and voters could be pre-registered by election officials. This is an example of policy rather than protocol; whatever security checks are necessary for registration occur at a higher level than the election itself.

The administrator may specify a time when the registration ends, or the collectors may manually force it to end using a menu option. Either way, when the registration closes, the collectors generate a share matrix for the location-vote and, based on the settings of the election, open either confirmation or voting.

Confirmation: This is an optional stage that serves to consolidate locations. From the voter's perspective, it is very simple: they enter their username and password and, assuming they registered successfully, they are shown a success message. However, the application quietly calculates the voter's absolute location and "votes" for it as discussed. The confirmation ends when all registered voters confirm or an optional timeout occurs, whichever comes first. The collectors tally the location vote exactly as they would a "real" vote, and publish it as a location vector.

If this behavior is not needed, the confirmation may be skipped with a setting in the election file. If pre-registration is in use, the confirmation may act as voter registration. For example, if a university wishes to conduct an election among its faculty, a list of qualified voters could be generated and the confirmation stage could be used to establish voter participation. Qualified but uninterested voters would not "register", and would be disqualified from the election after the timeout occurs.

Voting: Once the confirmation is finished, collectors compensate for any unconfirmed voters and open voting. Alternatively, if the confirmation is not being done, collectors may open voting immediately after the registration closes with no additional work done. During this stage, voters log in using their usernames and passwords and are presented with the names of the races and radio buttons corresponding to candidates, pictured in Fig. 8.3. Behind the scenes, the login procedure requests all necessary cryptographic data from the collectors, obtains the absolute location, and if the confirmation was done obtains the relative location as well. These mathematical processes may occur while the voter is making the choices. When the voter is finished, a voting vector is generated for each instance of each race, offset by location (relative if possible, absolute otherwise), encrypted with the corresponding keys, and sent to the collectors. When all voters have voted or an optional adminspecified timeout occurs, the collectors independently tally and publish the results and the election ends. Each collector should produce exactly the same tally results. If collectors disagree as to the outcome, misbehavior has occurred at certain level.

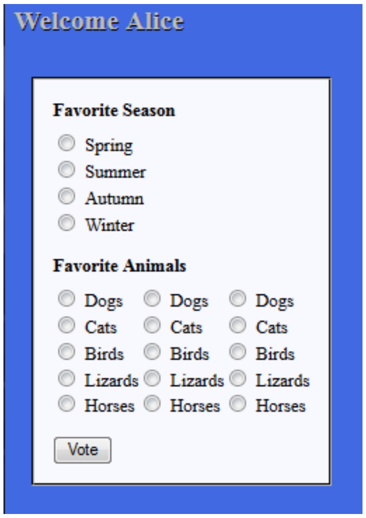


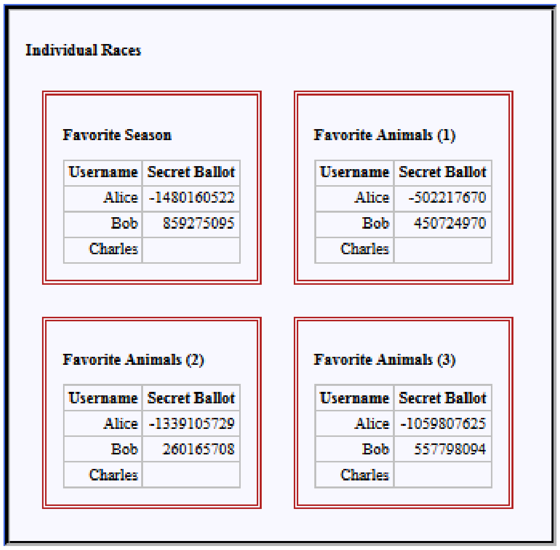
Fig. 8.3. Voting screenshot

8.3 Transparency

The collectors maintain two web pages for the publishing of data: first a raw results page that displays encrypted data in real time during the election, and secondly an interpreted results page that displays the tallied results after the election is over.

The raw results page is opened at the beginning of registration and acts as a dynamic bulletin board: during registration, it displays taken usernames as they are registered. During confirmation, it displays in yes-no format whether or not that voter has confirmed. During voting, it displays the encrypted ballots (numbers) for each instance of each race as they are cast. A snapshot of this page midway through the voting stage is pictured in Fig. 8.4.

The interpreted results page is opened at the close of the election when the ballots are tallied and acts as the actual results-viewer. At the top, it displays the winner(s) of



Individual Races

Favorite Season

Username	Secret Ballot
Alice	-1480160522
Bob	859275095
Charles	

Favorite Animals (1)

Username	Secret Ballot
Alice	-502217670
Bob	450724970
Charles	

Favorite Animals (2)

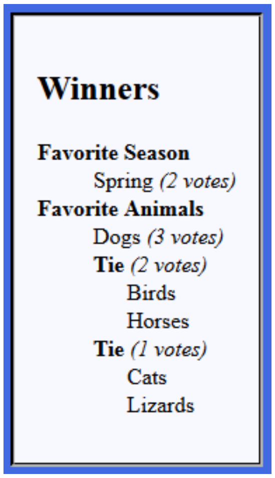
Username	Secret Ballot
Alice	-1339105729
Bob	260165708
Charles	

Favorite Animals (3)

Username	Secret Ballot
Alice	-1059807625
Bob	557798094
Charles	

Fig. 8.4. Raw results page: in-progress bulletin board

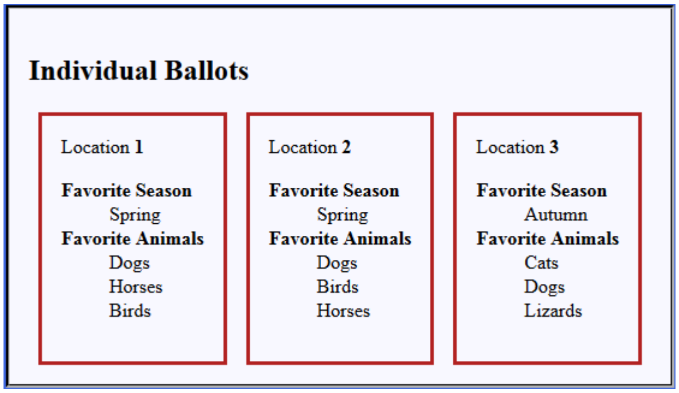
each race along with how many votes they obtained. Below, it displays each location and the choices that voter made. Examples of both are shown in Figs. 8.5 and 8.6. Thus, anyone may verify that the tallied results match perfectly the actual votes cast, and any voter may verify that their location's choices are exactly the choices they made. As the encrypted ballots are visible in the raw results page, anyone may run their own tally and verify that the interpreted locations accurately match what was cast. All of these verifications are done by simple observation that anyone, even non-technical people, can perform and understand.



Winners

Favorite Season
  Spring *(2 votes)*
Favorite Animals
  Dogs *(3 votes)*
  Tie *(2 votes)*
    Birds
    Horses
  Tie *(1 votes)*
    Cats
    Lizards

Fig. 8.5. Interpreted results page: displaying winners (including ties)



Individual Ballots

Location 1

Favorite Season
  Spring
Favorite Animals
  Dogs
  Horses
  Birds

Location 2

Favorite Season
  Spring
Favorite Animals
  Dogs
  Birds
  Horses

Location 3

Favorite Season
  Autumn
Favorite Animals
  Cats
  Dogs
  Lizards

Fig. 8.6. Interpreted results page: full content of all ballots

9. DISCUSSION OF SCALABILITY AND HIERARCHICAL DESIGN

In this chapter, we discuss scalability and design of our protocol, mainly for the non-interactive protocol.

Given the number of candidates M, the size of the voting vector determines the number of voters in one voting group and furthermore determines how large the integral vote values can be. Currently, most languages support arithmetical operations on big integers of arbitrary sizes. We conducted preliminary experiments with a voting group of 2000 voters and 2 candidates (i.e., voting vectors of 4000 bits) in Section 4.3.2. The results show an encouraging and impressive running time. Based on a 2004 survey by the US EAC on voting booth based elections, the average precinct size is approximately 1100 registered voters in the US 2004 presidential election [74]. Thus, our proposed voting system is realistic and practical. Furthermore, by following the US government structure and the precinct based election practice, we propose the following hierarchical tallying architecture which can apply to various elections of different scales.

- **Level 1**: Precinct based vote-casting. Voters belonging to a precinct form a voting group and execute the proposed vote-casting. Precincts may be based on the physical geography previously using voting booths or logically formed online to include geographically remote voters (e.g., overseas personnel in different countries).
- **Level 2**: Statewide vote tally. Perform anonymous tally among all precincts of a state.
- **Level 3**: Conversion of tallied votes. There can be a direct conversion. The numbers of votes for candidates from Level 2 remain unchanged and are passed

to Level 4 (the popular vote). Otherwise, they may be converted from Level 2 by some rules before being passed to Level 4, to support hierarchies like the Electoral Colleges in the US.

- **Level 4**: National vote tally.

10. RELATED WORK AND COMPARISON

Extensive research on voting, particularly online voting recently, has been conducted. A number of voting schemes and systems have been proposed [7–12, 17, 29, 68, 75–82].

10.1 Development of Voting Techniques

Cryptographic technique has been an indispensable component of most online voting systems. A variety of cryptographic techniques, such as mix-net, blind signature, homomorphic encryption, and secret sharing, are deployed in electronic voting protocols to secure voter's vote. The first e-voting scheme proposed by Chaum [83] in 1981 utilizes anonymous channels (i.e., mix-net). Additional schemes [7, 13, 84–89] based on mix-net are proposed afterwards with various optimization. For example, Aditya et al. [88] improve the efficiency of Lee et al.'s scheme [85] through modified optimistic mix-net. The scheme in [7] uses two kinds of mix-net to prevent vote updating from being detected by coercers. However, due to the usage of mix-net, transparency cannot be guaranteed.

A blind signature allows an authority to sign an encrypted message without knowing the message's context [14, 71, 89–95]. However, it is difficult to defend against misbehavior by authorities. Moreover, some participants (e.g., authorities) know intermediate results before the counting stage. This violates fairness of the voting protocol. Ring signature is proposed to replace the single signing authority. The challenge of using the ring signature is in preventing voters from double voting. Chow et al. [96] propose using a linkable ring signature, in which messages signed by the same member can be correlated, but not traced back to the member. A scheme combining blind signature and mix-net is proposed in [92]. Similarly, blind signature is used in a debate voting [95] where anonymous channel is assumed and messages are of varying length.

Voting schemes based on homomorphic encryption can trace back to the seminal works by Benaloh [16, 97], and later development in efficiency [53, 69] and receipt-freeness [15,52,98,99]. Rjaskova's scheme [15] achieves receipt-freeness by using deniable encryption, which allows a voter to produce a fake receipt to confuse the coercer. But eligibility and multi-voting prevention are not addressed. DEMOS-2 proposed by Kiayias et al. [100] utilizes additively homomorphic public keys on bilinear groups with the assumption that symmetric external Diffie-Hellman on these groups is hard. Its voting support machine (VSD) works as a "voting booth" and the voting protocol is rather complex.

Several voting schemes exploit homomorphism based on secret sharing [15,16,52, 97–99]. Some schemes [53, 69] utilize Shamir's threshold secret sharing [59], while some [54] are based on Chinese Remainder Theorem. In contrast, ours is based on a simplified (n, n)-SS scheme. In the existing voting schemes, secret sharing is utilized among authorities in two ways generally: a) to pool their shares together to get the vote decryption key which decrypts the tallied votes [15,16,52,53,97,98,101], and b) to pool their shares together to recover the encrypted or masked tally result [54, 69]. Instead, in our scheme, secret sharing is used among voters to share their secret votes and then recover their open yet anonymous votes.

Particularly, some existing voting protocols require physical settings such as voting booth [21], a tamper resistant randomizer [52, 85, 102, 103], or specialized smart cards [104]. Our protocol does not require specialized devices and is distributed by design.

We also examined experimental voting systems. Most existing systems have voter verifiability and usually provide vote anonymity and voter privacy by using cryptographic techniques. Typically, the clerks/officers at the voting places will check eligibility by verifying voters' identity.

Using the voting booth settings, system scalability in terms of voter numbers is hard to evaluate. Prêt à Voter [84, 105] encodes a voter's vote using a randomized candidate list. The randomization ensures the secrecy of a voter's vote. After casting his vote in a voting booth, a voter is given a receipt such that the voter can verify if his receipt appears on the bulletin board. Unlike our proposed protocol however, a voter will not see directly that his vote is counted. A number of talliers will recover the candidate list through the shared secret key and obtain the voter's vote.

ThreeBallot [106, 107] solves the verification and anonymity problem by giving each voter three ballots. The voter is asked to choose one of the three ballots to be verifiable. The ThreeBallot system requires a trusted authority to ensure that no candidate is selected on all three ballots to avoid multiple-vote fraud.

Punchscan/Scantegrity [108–111] allows the voter to obtain a confirmation code from the paper ballot. Through the confirmation code, the voter can verify if the code is correct for his ballot. Similarly to Prêt à Voter, a voter will not directly see that his vote is counted. A number of trustees will generate the tally which is publicly auditable.

SplitBallot [21] is a (physical) split ballot voting mechanism by splitting the trust between two conflict-of-interests parties or tallying authorities. It requires the untappable channels to guarantee everlasting privacy.

Prêt à Voter, Punchscan/Scantegrity, ThreeBallot, and SplitBallot utilize paper ballots and/or are based on voting booths, but ours does not. ThreeBallot and SplitBallot seem similar to ours in terms of split trust, however both of them depend on splitting paper ballots, unlike our protocol which utilizes electronic ballots that are split equally between two tallying collectors.

Bingo Voting [112] requires a random number list for each candidate which contains as many large random numbers as there are voters. In the voting booth, the system requires a random number generator.

VoteBox [113,114] utilizes a distributed broadcast network and replicated log, providing robustness and auditability in case of failure, misconfiguration, or tampering.

The system utilizes an immediate ballot challenge to assure a voter that his ballot is cast as intended. Additionally, the vote decryption key can be distributed to several mutually-untrusted parties. VoteBox provides strong auditing functionality but does not address how a voter can verify if his vote is really counted.

Prime III [115, 116] is a multimodal voting system especially devoted to the disabled and it allows voters to vote, review, and cast their ballots privately and independently through speech and/or touch. It offers a robust multimodal platform for the disabled but does not elaborate on how individual or universal verification is done.

Scytl [64, 117–119] requires dedicated hardware - a verification module (a physical device) on top of the DRE. Also, the trust, previously on the DRE, is transferred to the verification module. In contrast, ours is cost-efficient and does not require additional hardware device.

In the ADDER [120] system, a public key is set up for the voting system, and the private key is shared by a set of authorities. Each voter encrypts his vote using the public key. The encrypted vote and its zero-knowledge proof are published on the bulletin board. Due to the homomorphic property, the encrypted tally is obtained by multiplying all encrypted votes on the bulletin board. The authorities then work together to obtain the decrypted tally. ADDER [120] is similar to ours in terms of Internet based voting and split trust, yet ADDER does not provide a direct view for a voter to see if his vote is indeed counted in the tally.

DRE-i (Direct Recording Electronic with Integrity) voting protocol presented in [121] shares certain similarities with our protocol in the sense that both do not require authorities to tally votes and achieve transparency. DRE-i is designed specifically on top of the original DRE machines with the assumption of a supervised voting environment such as a polling station or voting booth. Instead, our protocol targets at Internet voting. Our protocol utilizes the conflicting nature in political parties, exploits the mathematically mutual restraining property, and develops effective sub-protocols to validate voters' shares and ballots in real-time during the voting process.

In addition, the DRE-i protocol currently is limited to “yes/no” voting, while our protocol supports not only “yes/no” voting but also voting for multiple candidates, and can be easily extended to support abstention (by adding one bit for each voter in the voting vector).

Due to the strict and conflicting e-voting requirements [5], unfortunately there is not any scheme/system currently satisfying all voting properties at the same time [81]. Security weakness is found even in highly referenced voting schemes [122]. The papers [123,124] particularly analyze two fundamental but important properties, privacy and verifiability. They reviewed the formal definitions of these two properties in the literature, and found that the scope and formulation of each property vary from one protocol to another. As a result, they propose a new game-based definition of privacy called BPRIV in [123] and a general definition of verifiability in [124].

Comparison with Helios. Helios [125] implements Benaloh’s vote-casting approach [126] on the Sako-Kilian mix-net [127]. It is a well-known and highly-accepted Internet voting protocol with good usability and operability. Our voting protocol shares certain features with Helios including open auditing and integrity.

However, there exist some important differences. *First*, about individual verification, Helios allows voters to verify their encrypted votes but our new protocol allows voters to verify their plain votes, in a visual manner. Thus, individual verification in the new protocol is more straightforward. *Second*, about transparency, as acknowledged by the author of Zeus, the mixing part in Helios (and Zeus) is a black box to voters [20]. Instead, in our protocol, the voting process including ballot-casting, ballot aggregation, plain vote verification, and tallying are all viewable (on public bulletin board) to voters. Thus, our protocol is more transparent. *Third*, in terms of voter assurance, the transition from ballots to plain votes in Helios involves mix-net (shuffling and reencryption) and decryption. In contrast, such transition in our protocol is seamless and viewable. In addition, a voter in our protocol can conduct self-tallying. Thus, voter assurance in our protocol is direct and personal. *Fourth*, about the trust issue (in terms of integrity of the tallying result), Helios depends on cryptographic techniques including zero knowledge proof to guarantee the trustworthiness of the mix-net which finally transforms to the integrity of the tallying result. In contrast, our protocol is straightly based on simple addition and viewable verification. Thus, accuracy of the tallying result in our protocol is obvious and is easier to justify. *Fifth*, about the trust issue (in terms of vote secrecy), Helios can use two or more mix-servers to split trust. However, it assumes that at least a certain number of mix-servers do not collude. In this case, it is similar to our assumption that two or more collectors have conflicting interests and will not collude. *Sixth*, about computational complexity, Helios' ballot preparation requires modular exponentiations for each voter and the tallying process also involves exponentiations (decryption). Instead, our ballot generation and tallying need only modular subtractions and additions. Thus, our protocol is more efficient.

Besides Zeus [20] and Helios 2.0 [128], there are some variants of Helios such as BeleniosRF [129]. BeleniosRF is built upon Belenios [130]. It introduces signatures on randomizable ciphertexts to achieve receipt-freeness. A voting authority is assumed to be trustworthy.

Comparison with Trivitas/Civitas. Trivitas [131], based on Civitas [132], also shares similarity with our protocol in the sense that both publish votes in plain text onto a bulletin board so a voter can visually check the board to verify his vote. The aforementioned gap is thus eliminated. However, there exists some differences. In Trivitas, trial credential and trial vote are introduced. Trial votes are decrypted and published as plain text on the bulletin board. A voter is assured through visual checking that the trial vote in plain text on the bulletin board is indeed what he has cast. Our protocol implements this from a different perspective. It does not need any decryption by authorities. After vote casting, a voter can simply aggregate all secret ballots to reveal all individual votes. He can then verify by visual checking that the vote in his location is indeed his vote.

Comparison with existing interactive voting protocols. In aforementioned voting protocols, most are centralized. Our non-interactive protocol is similar in this regard. However, some e-voting protocols are decentralized or distributed: each voter bears the same load, executes the same protocol, and reaches the same result autonomously [133]. One interesting application domain of distributed e-voting is boardroom voting [18]. In such scenario, the number of voters is not large and all the voters in the voting process will interact with each other. Two typical boardroom voting protocols are the ones in [19,134] and our interactive voting protocol is similar to them. In all these protocols including ours, tallying is called self-tallying: each voter can incrementally aggregate ballots/votes themselves by either summing the votes [134]) (as well as ours) or multiplying the ballots [19]) (and then verify the correctness of the final tally). One main advantage of our interactive voting protocol over other distributed e-voting protocols is its low computation cost for vote casting and tallying. As analyzed in [18], in terms of vote casting, the protocol in [19] needs $4N + 10$ exponentiations per voter and the protocol in [134] needs $2N + 2$ exponentiations. However, our interactive voting protocol needs only $2N - 2$ modular additions/subtractions. In terms of tallying [18], the protocol in [19] needs $11N - 11 + \binom{N+M}{M}/2$ and the protocol in [134] needs $19N^2 + 58N - 14Nt - 17t + 35$ (here t is the threshold in distributed key generation scheme). However, our interactive voting protocol does not need any exponentiations besides simple modular additions. Another property of our interactive voting protocol in terms of transparency is the viewability of the voter's plain vote: each voter knows and can see which plain vote is his vote. However, in [19], plain votes are not viewable, and in [134], even though plain votes are viewable but the voter does not know which one is his vote because the shuffling process changes the correspondence between the initial encrypted ballots and individual plain votes. On the other hand, our interactive voting protocol needs the involvement of collectors which check and enforce the protocol compliance from each voter, which increases the overall protocol cost.

10.2 Receipt-freeness and Coercion-resistance

Receipt-freeness refers to a scenario where a voter cannot prove to a third party that he has cast vote for a particular candidate. It is first proposed by Benaloh and Tuinstra in [68]. Jonker et al. [135] give an informal definition of receipt and list its properties. However, without certain assumptions, there is incompatibility between receipt-freeness and universal verifiability [136]. Thus, secret channels between voters and authorities are usually assumed in order to achieve both.

Coercion-resistance deals with more powerful adversaries. The concept was first discussed in [33], and formally introduced by Juels et al. in [13] for e-voting. Generally speaking, an adversary may coerce a targeted voter into casting vote in a particular manner, abstaining from voting, or even giving in his secret keys. A coercion-resistant scheme is capable of circumventing the adversary's demands so that the adversary cannot distinguish whether a coerced voter follows his dictations.

The definition of coercion-resistance has been explored by several research groups. Juels et al. [13] take an algorithmic approach. Their algorithms simulate two experiments of coercion-resistance, with the first characterizing an adversary capable of distinguishing the fake key from the genuine private key (complete coercion of the targeted voter), and the second characterizing an ideal situation where the adversary cannot determine whether the coercion is successful.

The definition in [137] takes an epistemic approach with symbolic settings. Küsters and Truderung also apply the framework of their definition to other e-voting models, and analyze the positive and negative aspects. The definitions given in [138–140] are game-based, which share certain similarity with [13]. However, as claimed in [138], their definition is more general and fits more e-voting protocols, while the definition in [13] is tailored for voting in a public-key setting, with protocols having a specific structure.

Delaune et al. [141] use the applied pi calculus to formalize coercion-resistance and receipt-freeness. From their formalization, coercion-resistance implies receipt-freeness, while receipt-freeness in turn implies privacy. They apply this formalization onto Lee et al.'s protocol [41], and model the initialization process, the keying material generation, and the distribution process, together with processes for each entity such as administrator, collector, and voter.

Moran and Naor [142] give a rather general definition of coercion-resistance within the simulation-based approach, extending from a definition of coercion-resistance in [33] for incoercible multiparty computation.

Recently, Clark and Hengartner [70] propose an over-the-shoulder coercion-resistant voting protocol called Selections. It is based on panic passwords. When a voter is coerced to cast a vote at the coercer's presence, the voter can cast the coerced vote using a panic password and then later, the voter will cast his real vote using his real password. Selections employs an anonymous channel or mix-net for vote anonymity and homomorphic encryption for tallying. Like most other coercion-resistant protocols, Selections does not have a clear mechanism for voters to verify their vote.

The recent research by Grewal et al. [6] acknowledges the toughness of the voter-coercion issue, and proposes to redefine its meaning and scope. Even for the century-old Australian ballot which used to be coercion-free, voter-coercion is becoming an issue because of new emerging video technology [143].

Table 10.1 gives a comparison of the voting schemes mentioned above.

Table 10.1.
Comparison of voting schemes

Schemes/ Systems	Individual Verifiability	Universal Verifiability	Fair-ness	Coercion-resistance	Robust-ness	Receipt freeness	Cryptographic primitives	Tailored Hardware
UVote [144]	Y	NK	NK	Y	NK	N	Mix-net	N
Zeus [145]	Y	Y	Y	N	Y	N	Mix-net, ZKP	N
Cobra [140]	N	N	Y	Y	Y	Y	HE, EBF	N
Helios [146]	Y	Y	Y	N	Y	N	Mix-net, ZKP	N
DSA Public Keys [147]	Y	Y	Y	Y	Y	NK	HE, Mix-net, ZPK	N
RSA Puzzle Lock [32]	Y	NK	Y	NK	Y	NK	RSA	N
Civitas [132]	Y	NK	Y	Y	Y	Y	Mix-net	N
Multi-Authority [148]	NK	Y	Y	Y	Y	Y	HE, ElGamal DSA	N
E-NOTE [149]	Y	Y	Y	NK	Y	N	RSA	Watchdog
Bingo [150]	Y	Y	Y	Y	Y	Y	CM, ZKP	Booth
VoteBox [151]	Y	Y	NK	Y	Y	Y	HE, HC	Booth
Prêt à Voter [152]	Y	Y	Y	Y	Y	Y	Mix-net	Booth
ADDER System [153]	NK	Y	Y	Y	Y	Y	HE, ZKP	N
Scytl [64]	Y	Y	Y	Y	Y	Y	SS, Mix-net	Booth, VM
Our protocol	Y	Y	Y	Y	Y	Y	SS	N

Y: Yes; N: No; NK: Not Known; HE: Homomorphic Encryption; EBF: Encrypted Bloom Filter; ZKP: Zero Knowledge Proof;
CM: Commitment; HC: Hash Chaining; SS: Secret Sharing; VM: Verification Module; Booth: Voting Booth

11. CONCLUSION AND FUTURE WORK

11.1 Conclusion

We proposed a fully transparent, auditable, and end-to-end verifiable voting protocol to enable open and fair elections. It exploits the conflicts of interest in multiple tallying authorities, such as the two-party political system in the US. Our protocol is built upon three novel technical contributions—verifiable voting vector, forward and backward mutual lock voting, and proven in-process check and enforcement. These three technical contributions, along with transparent vote casting and tallying processes, incremental aggregation of secret ballots, and incremental vote tallying for candidates, deliver fairness and voter assurance. Each voter can be assured that his vote is counted both technically and visually.

In addition, for the voting environment requiring receipt-freeness and coercion-resistance, we introduced solutions on top of the voting protocol to counter vote selling or buying with receipt and voter coercion.

In particular, the interactive protocol is suitable for election within a small group such as boardroom voting where interaction is encouraged, while the non-interactive protocol is designed for election within a large group where interaction is not needed and not realistic. Through the analysis and simulation, we demonstrated the robustness, effectiveness, and feasibility of our voting protocol.

11.2 Future Work

First, we will investigate and extend our e-voting protocol to other voting scenarios such as write-in candidate voting.

Second, we will further explore location anonymization schemes (LAS). Currently our original LAS requires multiple rounds of computation in order for each voter to get a unique location; while the CRT based LAS requires a very large location vector. we need to find a LAS which is more efficient in both computation and space.

Third, we will further look into the vote-selling and voter-coercion issues. Our current solutions are far from perfect, so there are lots of potentials to improve our solutions.

Fourth, with better cryptanalysis techniques being developed and the emergence of quantum computers, the discrete logarithm problem (DLP), which our protocol relies on, will not be hard any more in the near future, consequently, the privacy of votes in our protocol can be compromised. We should also investigate whether everlasting privacy can be incorporated into our protocol.

REFERENCES

REFERENCES

[1] Wikipedia, "Help america vote act, the free encyclopedia," 2013, [Online; accessed 1-July-2013]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Help_America_Vote_Act&oldid=543758991

[2] ——, "The U.S. Election Assistance Commission," 2013, [Online; accessed 1-July-2013]. [Online]. Available: http://www.eac.gov/

[3] T. U. E. A. Commission, "2008 election administration and voting survey," http://www.eac.gov/, 2009.

[4] B. Campbell, "The usability implications of long ballot content for paper, electronic, and mobile voting systems," Ph.D. dissertation, RICE Uni., 2014.

[5] B. Chevallier-Mames, P. A. Fouque, D. Pointcheval, J. Stern, and J. Traoré, "On some incompatible properties of voting schemes. (book chapter)," in *Towards Trustworthy Elections*, D. Chaum, M. Jakobsson, R. L. Rivest, P. A. Ryan, and J. Benaloh, Eds., 2010, pp. 191–199.

[6] G. S. Grewal, M. D. Ryan, S. Bursuc, and P. Y. Ryan, "Caveat coercitor: coercion-evidence in electronic voting," in *IEEE Symposium on Security and Privacy*, 2013, pp. 367–381.

[7] R. Araújo, A. Barki, S. Brunet, and J. Traoré, *Remote Electronic Voting Can Be Efficient, Verifiable and Coercion-Resistant.* Springer Berlin Heidelberg, 2016, pp. 224–232.

[8] C. Li, M. Hwang, and Y. Lai, "A verifiable electronic voting scheme over the internet," in *ITNG '09*, april 2009, pp. 449 –454.

[9] J. Howlader, V. Nair, S. Basu, and A. Mal, "Uncoercibility in e-voting and e-auctioning mechanisms using deniable encryptiony," *International Journal of Network Security and Its Applications*, vol. 3, no. 2, pp. 97–109, 2011.

[10] C. Lee, T. Chen, S. Lin, and M. Hwang, "A new proxy electronic voting scheme based on proxy signatures," ser. Lecture Notes in Electrical Engineering, C.-L. W. J. Park, V. C.M. Leung and T. Shon, Eds., 2012, vol. 164, pp. 3–12.

[11] O. Spycher, R. Koenig, R. Haenni, and M. Schlpfer, "A new approach towards coercion-resistant remote e-voting in linear time," ser. LNCS, 2012, vol. 7035, pp. 182–189.

[12] M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a secure voting system," in *Proc. of IEEE S & P*, 2008, pp. 354–368.

[13] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proc. of ACM WPES*. ACM, 2005, pp. 61–70. [Online]. Available: http://doi.acm.org/10.1145/1102199.1102213

[14] D. Chaum, "Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa," in *EUROCRYPT'88*, 1988, pp. 177–182.

[15] Z. Rjaskova, "Electronic voting schemes," Ph.D. dissertation, Comenius Uni., 2002.

[16] J. Benaloh, "Verifiable secret ballot elections," Ph.D. dissertation, Yale Uni., 1987.

[17] D. Jones and B. Simons, *Broken Ballots: Will Your Vote Count?* The University of Chicago Press, June 2012.

[18] O. Kulyk, S. Neumann, J. Budurushi, M. Volkamer, R. Haenni, R. Koenig, and P. Bergen, "Efficiency evaluation of cryptographic protocols for boardroom voting," *2015 10th International Conference on Availability, Reliability and Security (ARES)*, pp. 224–229, 2015.

[19] D. Khader, B. Smyth, P. Y. Ryan, and F. Hao, "A fair and robust voting system by broadcast," *EVOTE12: 5th International Conference on Electronic Voting*, pp. 1–13, 2012.

[20] G. Tsoukalas, K. Papadimitriou, P. Louridas, and P. Tsanakas, "From helios to zeus," in *EVT/WOTE*. USENIX, 2013, pp. 1–17. [Online]. Available: https://www.usenix.org/conference/evtwote13/workshop-program/presentation/Tsoukalas

[21] T. Moran and M. Naor, "Split-ballot voting: Everlasting privacy with distributed trust," *ACM Trans. Inf. Syst. Security*, vol. 13, no. 2, pp. 16:1–16:43, 2010. [Online]. Available: http://doi.acm.org/10.1145/1698750.1698756

[22] Wikipedia, "Vote counting system," 2012, [Online; accessed 21-Jun.-2012]. [Online]. Available: http://en.wikipedia.org/wiki/Vote_counting_systems

[23] H. Li, A. R. Kankanala, and X. Zou, "A taxonomy and comparison of remote voting schemes," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–8.

[24] M. Volkamer and R. Grimm, "Determine the resilience of evaluated internet voting systems," in *Requirements Engineering for e-Voting Systems (RE-VOTE), 2009 First International Workshop on*, aug. 2009, pp. 47–54.

[25] C.-T. Li and M.-S. Hwang, "Security enhancement of chang-lee anonymous e-voting scheme," *International Journal of Smart Home*, vol. 6, no. 2, pp. 45–52, April 2012.

[26] J. Epstein, "Electronic voting," *IEEE Computer*, vol. 40, no. 8, pp. 92–95, 2007.

[27] T. Wan and W. Liao, "Cryptanalysis on polynomial-based e-voting schemes," in *2010 International Conference on E-Business and E-Government (ICEE)*, 2010, pp. 436–438.

[28] R. Rivest, “On the notion of software independence in voting systems,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3759–3767, 2008.

[29] C. Staff, “Seven principles for secure e-voting,” *Commun. ACM*, vol. 52, no. 2, pp. 8–9, Feb. 2009. [Online]. Available: http://doi.acm.org/10.1145/1461928.1461931

[30] S. Yun and S. Lee, “The network based electronic voting scheme suitable for large scale election,” in *The 6th International Conference on Advanced Communication Technology*, vol. 1, 2004, pp. 218–222.

[31] O. Cetinkaya and A. Doganaksoy, “A practical verifiable e-voting protocol for large scale elections over a network,” in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, 2007, pp. 432–442.

[32] H. Chen and R. Deviani, “A secure e-voting system based on rsa time-lock puzzle mechanism,” in *The 7th International Conference on BWCCA*, 2012, pp. 596–601.

[33] R. Canetti and R. Gennaro, “Incoercible multiparty computation,” in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, ser. FOCS ’96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 504–. [Online]. Available: http://dl.acm.org/citation.cfm?id=874062.875484

[34] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, “Deniable encryption,” in *Advances in Cryptology CRYPTO ’97*, ser. Lecture Notes in Computer Science, B. Kaliski, Ed. Springer Berlin / Heidelberg, 1997, vol. 1294, pp. 90–104, 10.1007/BFb0052229. [Online]. Available: http://dx.doi.org/10.1007/BFb0052229

[35] M. Klonowski, P. Kubiak, and M. Kutyłowski, “Practical deniable encryption,” in *SOFSEM 2008: Theory and Practice of Computer Science*, ser. Lecture Notes in Computer Science, V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, Eds. Springer Berlin / Heidelberg, 2008, vol. 4910, pp. 599–609, 10.1007/978-3-540-77566-9_52. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77566-9_52

[36] A. C.-C. Yao, “How to generate and exchange secrets,” in *Foundations of Computer Science, 1986., 27th Annual Symposium on*, Oct. 1986, pp. 162–167.

[37] S. Samet and A. Miri, “Privacy preserving ID3 using Gini index over horizontally partitioned data,” ser. AICCSA ’08, 2008, pp. 645–651. [Online]. Available: http://dx.doi.org/10.1109/AICCSA.2008.4493598

[38] M. Jakobsson and A. Juels, “Mix and match: Secure function evaluation via ciphertexts,” in *Advances in Cryptology ASIACRYPT 2000*, ser. Lecture Notes in Computer Science, T. Okamoto, Ed. Springer Berlin / Heidelberg, 2000, vol. 1976, pp. 162–177, 10.1007/3-540-44448-3_13. [Online]. Available: http://dx.doi.org/10.1007/3-540-44448-3_13

[39] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981. [Online]. Available: http://doi.acm.org/10.1145/358549.358563

[40] B. Adida, "Advances in cryptographic voting systems," Ph.D. dissertation, Cambridge, MA, USA, 2006, aAI0810143.

[41] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo, "Providing receipt-freeness in mixnet-based voting protocols," in *In Proc. of Information Security and Cryptology (ICISC03), volume 2971 of LNCS*. Springer, 2003, pp. 245–258.

[42] P. Ribarski and L. Antovski, "Mixnets: Implementation and performance evaluation of decryption and re-encryption types," in *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on*, 2012, pp. 493–498.

[43] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of CRYPTO '82*. Plenum, 1982, pp. 199–203.

[44] S. Goldwasser and M. Bellare, "Lecture notes on cryptography," *Summer course Cryptography and computer security at MIT*, 1999.

[45] J. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract)," in *Advances in Cryptology - CRYPTO'86*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1987, pp. 251–260. [Online]. Available: http://dx.doi.org/10.1007/3-540-47721-7_19

[46] J. Benaloh and M. Yung, "Distributing the power of a government to enhance the privacy of voters," in *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, ser. PODC '86. New York, NY, USA: ACM, 1986, pp. 52–62. [Online]. Available: http://doi.acm.org/10.1145/10590.10595

[47] Wikipedia, "Homomorphic encryption," 2012, [Online; accessed 21-October-2012]. [Online]. Available: \url{http://en.wikipedia.org/wiki/Homomorphic_encryption}

[48] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*, ser. Lecture Notes in Computer Science, G. Blakley and D. Chaum, Eds. Springer Berlin / Heidelberg, 1985, vol. 196, pp. 10–18. [Online]. Available: http://dx.doi.org/10.1007/3-540-39568-7_2

[49] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology EUROCRYPT 99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin / Heidelberg, 1999, vol. 1592, pp. 223–238, 10.1007/3-540-48910-X_16. [Online]. Available: http://dx.doi.org/10.1007/3-540-48910-X_16

[50] E. Dawson and D. Donovan, "The breadth of shamir's secret-sharing scheme," *Comput. Secur.*, vol. 13, no. 1, pp. 69–78, Feb. 1994.

[51] X. Zhao, L. Li, G. Xue, and G. Silva, "Efficient anonymous message submission," 2012, pp. 2228–2236.

[52] B. Lee and K. Kim, "Receipt-free electronic voting through collaboration of voter and honest verifier," in *Proc. of JW-ISC*, 2000, pp. 101–108.

[53] R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” ser. EUROCRYPT’97, 1997, pp. 103–118. [Online]. Available: http://dl.acm.org/citation.cfm?id=1754542.1754554

[54] S. Iftene, “General secret sharing based on the chinese remainder theorem with applications in e-voting,” *Electron. Notes Theor. Comp. Sci.*, vol. 186, pp. 67–84, Jul. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2007.01.065

[55] K. Sampigethaya and R. Poovendran, “A survey on mix networks and their secure applications,” *Proceedings of the IEEE*, vol. 94, no. 12, pp. 2142–2181, 2006.

[56] J. Esch, “Prolog to a survey on mix networks and their secure applications,” *Proceedings of the IEEE*, vol. 94, no. 12, pp. 2139–2141, 2006.

[57] H. Li, Y. Sui, W. Peng, X. Zou, and F. Li, “A viewable e-voting scheme for environments with conflict of interest,” in *2013 IEEE Conference on Communications and Network Security (CNS)*, Oct 2013, pp. 251–259.

[58] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications).* Chapman & Hall/CRC, Nov 2005.

[59] A. Shamir, “How to share a secret,” *Comm. of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[60] D. R. Stinson, *Cryptography: Theory and Practice.* CRC Press, 1995.

[61] X. Zhao, L. Li, G. Xue, and G. Silva, “Efficient anonymous message submission,” in *Proc. of IEEE INFOCOM*, May 2012, pp. 2228–2236.

[62] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *28th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct. 1987, pp. 427–438.

[63] T. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in *Advances in Cryptology - CRYPTO’91*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1992, pp. 129–140. [Online]. Available: http://dx.doi.org/10.1007/3-540-46766-1_9

[64] M. Clarkson, B. Hay, M. Inge, D. Wagner, and A. Yasinsac, “Software review and security analysis of scytl remote voting software,” Sep. 2008.

[65] X. Zou, H. Li, Y. Sui, W. Peng, and F. Li, “Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties,” in *Proc. of IEEE INFOCOM*, 2014, pp. 136–144.

[66] X. Zou, H. Li, F. Li, W. Peng, and Y. Sui, “Transparent, auditable, and stepwise verifiable online e-voting enabling an open and fair election,” *Cryptography*, vol. 1, no. 2, 2017.

[67] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644 – 654, Nov 1976.

[68] J. Benaloh and D. Tuinstra, “Receipt-free secret-ballot elections,” in *Proceedings of the 26th Annual ACM Symposium on Theory of Computing.* New York: ACM, 1994, pp. 544–553.

[69] B. Schoenmakers, “A simple publicly verifiable secret sharing scheme and its application to electronic voting,” ser. CRYPTO’99, 1999, pp. 148–164.

[70] J. Clark and U. Hengartner, “Selections: Internet voting with over-the-shoulder coercion-resistance,” in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, G. Danezis, Ed. Springer Berlin / Heidelberg, 2012, vol. 7035, pp. 47–61.

[71] A. Fujioka, T. Okamoto, and K. Ohta, “A practical secret voting scheme for large scale elections,” in *Advances in Cryptology-AUSCRYPT’92*. Springer, 1993, pp. 244–251.

[72] E. H. Spafford, “Voter assurance.” *Voting Technologies, National Academy of Engineering*, pp. 28 –34, 2007.

[73] K. Butterfield, H. Li, X. Zou, and F. Li, “Enhancing and implementing fully transparent internet voting,” in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, Aug 2015, pp. 1–6.

[74] T. U. E. A. Commission, “Polling places 2004 general election: Eac election day survey.” 2006. [Online]. Available: http://web.archive.org/web/20061214025307/http://www.eac.gov/election_survey_2004/chapter_table/Chapter13\\\_Polling_Places.htm

[75] M. Clarkson, S. Chong, and A. Myers, “Civitas: Toward a secure voting system,” *Technical Report*, May 2008, cornell University.

[76] F. Shirazi, S. Neumann, I. Ciolacu, and M. Volkamer, “Robust electronic voting: Introducing robustness in civitas,” in *International Workshop on REVOTE*, Aug. 2011, pp. 47 –55.

[77] M. Volkamer and R. Grimm, “Determine the resilience of evaluated internet voting systems,” in *International Workshop on REVOTE*, Aug. 2009, pp. 47 –54.

[78] C. Li and M. Hwang, “Security enhancement of Chang-Lee anonymous e-voting scheme,” *Int. Jour. of Smart Home*, vol. 6, no. 2, pp. 45–52, 2012.

[79] J. Epstein, “Electronic voting,” *IEEE Computer*, no. 40, pp. 92–95, 2007.

[80] R. Kusters, T. Truderung, and A. Vogt, “Clash attacks on the verifiability of e-voting systems,” in *IEEE S & P*, may 2012, pp. 395–409.

[81] L. Fouard, M. Duclos, and P. Lafourcade, “Survey on electronic voting schemes,” http://www-verimag.imag.fr/ duclos/paper/e-vote.pdf.

[82] J. Benaloh, M. Byrne, P. Kortum, N. McBurnett, O. Pereira, P. Stark, and D. Wallach, “Star-vote: A secure, transparent, auditable, and reliable voting system,” *arXiv preprint arXiv:1211.1904*, 2012.

[83] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Comm. of the ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.

[84] D. Chaum, P. Ryan, and S. Schneider, “A practical voter-verifiable election scheme,” in *Proc. of the 10th European Conf. on Research in Comp. Security*, ser. ESORICS’05, Milan, Italy, 2005, pp. 118–139.

[85] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo, "Providing receipt-freeness in mixnet-based voting protocols," in *Proc. of Info. Security and Cryptology*, vol. 2971, 2003, pp. 245–258.

[86] S. Weber, "A coercion-resistant cryptographic voting protocol - evaluation and prototype implementation," *Master's thesis, Darmstadt University of Technology*, 2006.

[87] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," *Security Privacy, IEEE*, vol. 2, no. 1, pp. 38 – 47, jan.-feb. 2004.

[88] R. Aditya., B. Lee, C. Boyd, and E. Dawson, "An efficient mixnet-based voting scheme providing receipt-freeness," ser. LNCS, 2004, vol. 3184, pp. 152–161.

[89] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *Security Protocols*. Springer, 1998, pp. 25–35.

[90] M. Radwin, "An untraceable, universally verifiable voting scheme," *Seminar in Cryptology*, 1995.

[91] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto, "An improvement on a practical secret voting scheme," in *Proceedings of the Second International Workshop on Information Security*, ser. ISW '99. London, UK, UK: Springer-Verlag, 1999, pp. 225–234.

[92] ——, "An improvement on a practical secret voting scheme," in *Proc. of 2nd Int. Workshop on Info. Security*, ser. ISW '99, 1999, pp. 225–234. [Online]. Available: http://dl.acm.org/citation.cfm?id=648023.744186

[93] K. Kim, J. Kim, B. Lee, and G. Ahn, "Experimental design of worldwide internet voting system using pki," in *Proc. of SSGRR*, 2001, pp. 1–7.

[94] C. Boyd, "A new multiple key cipher and an improved voting scheme," ser. EUROCRYPT '89, 1990, pp. 617–625.

[95] D. L. García, "A flexible e-voting scheme for debate tools," *Computers & Security*, vol. 56, pp. 50 – 62, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404815001546

[96] S. Chow, J. Liu, and D. Wong, "Robust receipt-free election system with ballot secrecy and verifiability," in *Proc. of NDSS*, 2008, pp. 81–94.

[97] J. Benaloh and M. Yung, "Distributing the power of a government to enhance the privacy of voters," ser. PODC '86, 1986, pp. 52–62.

[98] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Proc. of the 19th Int. Conf. on Theory and App. of crypto. techniques*, 2000, pp. 539–556.

[99] A. P. Adewole, A. S. Sodiya, and O. A. Arowolo, "A receipt-free multi-authority e-voting system," *Int. Jour. of Comp. App.*, vol. 30, no. 6, pp. 15–23, 2011.

[100] T. Kiayias, A.and Zacharias and B. Zhang, "Demos-2: Scalable e2e verifiable elections without random oracles," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 352–363.

[101] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, "Multi-authority secret-ballot elections with linear work," in *Advances in Cryptology EUROCRYPT 96*, 1996, vol. 1070, pp. 72–83. [Online]. Available: http://dx.doi.org/10.1007/3-540-68339-9_7

[102] O. Baudron, P. A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proc. of 20th ACM Sym. on Principles of distributed computing*, ser. PODC '01, 2001, pp. 274–283.

[103] M. Hirt, "Receipt-free k-out-of-l voting based on elgamal encryption," in *Towards Trustworthy Elections*, D. Chaum, M. Jakobsson, R. L. Rivest, P. A. Ryan, and J. Benaloh, Eds., 2010, pp. 64–82.

[104] W. Han, K. Chen, and D. Zheng, "Receipt-freeness for groth's e-voting schemes," *J. of Information Science and Engineering*, no. 25, pp. 517–530, 2009.

[105] P. Ryan and T. Peacock, "Prêt à voter: a systems perspective," *University of Newcastle, Technical Report CS-TR-929*, 2005.

[106] R. Rivest, "The threeballot voting system," *Avaialable at: http://theory. lcs. mit. edu/rivest/Rivest-TheThreeBallotVotingSystem. pdf*, 2006.

[107] R. Rivest and W. Smith, "Three voting protocols: Threeballot, vav, and twin," in *Proc. of the USENIX AEVT*, vol. 16, 2007, pp. 1–14.

[108] K. Fisher, R. Carback, and A. Sherman, "Punchscan: Introduction and system definition of a high-integrity election system," in *Proceedings of Workshop on Trustworthy Elections*, 2006, pp. 19–29.

[109] S. Popoveniuc and B. Hosp, "An introduction to punchscan," in *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, 2006, pp. 28–30.

[110] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora, "Scantegrity: End-to-end voter-verifiable optical-scan voting," *Security & Privacy, IEEE*, vol. 6, no. 3, pp. 40–46, 2008.

[111] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. Rivest, P. Ryan, E. Shen, and A. Sherman, "Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes," *EVT*, vol. 8, pp. 1–13, 2008.

[112] J. Bohli, J. Müller-Quade, and S. Röhrich, "Bingo voting: Secure and coercion-free voting using a trusted random number generator," *E-Voting and Identity*, pp. 111–124, 2007.

[113] D. Sandler and D. Wallach, "Casting votes in the auditorium," in *Proc. of AEVT*, 2007, pp. 1–15.

[114] D. Sandler, K. Derr, and D. Wallach, "Votebox: a tamper-evident, verifiable electronic voting system," in *USENIX Security Symposium*, vol. 4, no. 0, 2008, pp. 349–364.

[115] E. V. Cross II, Y. McMillian, P. Gupta, P. Williams, K. Nobles, and J. E. Gilbert, "Prime III: a user centered voting system," in *CHI*, 2007, pp. 2351–2356.

[116] S. Dawkins, T. Sullivan, G. Rogers, E. V. C. II, L. Hamilton, and J. E. Gilbert, "Prime iii: an innovative electronic voting interface," in *IUI*, 2009, pp. 485–486.

[117] Scytl, "White paper: Auditability and voter-verifiability for electronic voting terminals," http://www.scytl.com/images/upload/home/PNYX.VM_White_Paper.pdf.

[118] SCYTL, "Pnyx.vm: Auditability and voter-verifiability for electronic voting terminals," in *White Paper*, 2004.

[119] ——, "Pnyx.core: The key to enabling reliable electronic elections," in *White Paper*, 2005.

[120] A. Kiayias, M. Korman, and D. Walluck, "An internet voting system supporting user privacy," in *Computer Security Applications Conference. ACSAC'06. 22nd Annual.* IEEE, 2006, pp. 165–174.

[121] F. Hao, M. N. Kreeger, B. Randell, D. Clarke, S. F. Shahandashti, and P. Lee, "Every vote counts: Ensuring integrity in large-scale electronic voting," *The USENIX Journal of Election Technology and Systems*, pp. 1–25, 2014.

[122] C. Karlof, N. Sastry, and D. Wagner, "Cryptographic voting protocols: A systems perspective," in *USENIX Security*, vol. 5, 2005, pp. 33–50.

[123] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "Sok: A comprehensive analysis of game-based ballot privacy definitions," in *IEEE Symposium on Security and Privacy.* IEEE, 2015, pp. 499–516.

[124] V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung, "Sok: Verifiability notions for e-voting protocols," in *36th IEEE Symposium on Security and Privacy (S&P'16)*, 2016, pp. 779–798.

[125] B. Adida, "Helios: Web-based open-audit voting," in *USENIX Security Symposium*, vol. 17. USENIX Association, 2008, pp. 335–348.

[126] J. Benaloh, "Simple verifiable elections," in *EVT/WOTE.* USENIX, 2006, pp. 1–10.

[127] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme," in *Advances in Cryptology-UROCRYPT'95.* Springer, 1995, pp. 393–403.

[128] B. Adida, O. De Marneffe, O. Pereira, and J. Quisquater, "Electing a university president using open-audit voting: Analysis of real-world use of helios," in *Proceedings of the Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2009.

[129] V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A strongly receipt-free electronic voting scheme," Cryptology ePrint Archive, Report 2015/629, 2015, http://eprint.iacr.org/2015/629.

[130] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Election verifiability for helios under weaker trust assumptions," in *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Springer International Publishing, 2014, pp. 327–344.

[131] S. Bursuc, G. S. Grewal, and M. D. Ryan, "Trivitas: Voters directly verifying votes," in *E-Voting and Identity*. Springer, 2012, pp. 190–207.

[132] M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a secure voting system," in *Proc. of IEEE S & P*, 2008, pp. 354–368.

[133] A. Nakajima, "Decentralized voting protocols," in *Proc. of Int. Sym. on Autonomous Decentralized Systems*. IEEE, 1993, pp. 247–254.

[134] O. Kulyk, S. Neumann, C. Feier, M. Volkamer, and T. Koster, "Electronic voting with fully distributed trust and maximized flexibility regarding ballot design," *6th International Conference on Electronic Voting (EVOTE). IEEE*, pp. 1–10, 2014.

[135] H. Jonker and E. de Vink, "Formalising receipt-freeness," in *Information Security*, ser. Lecture Notes in Computer Science, S. Katsikas, J. López, M. Backes, S. Gritzalis, and B. Preneel, Eds. Springer Berlin / Heidelberg, 2006, vol. 4176, pp. 476–488, 10.1007/11836810_34. [Online]. Available: http://dx.doi.org/10.1007/11836810_34

[136] B. Chevallier-Mames, P. A. Fouque, D. Pointcheval, and J. Traoré, "On some incompatible properties of voting schemes," in *WOTE 06*, 2006.

[137] R. Küsters and T. Truderung, "An epistemic approach to coercion-resistance for electronic voting protocols," in *Security and Privacy, 2009 30th IEEE Symposium on*, May 2009, pp. 251–266.

[138] R. Küsters, T. Truderung, and A. Vogt, "A game-based definition of coercion-resistance and its applications," in *Proceedings of the 2010 23rd IEEE Computer Security Foundations Symposium*. IEEE Computer Society, 2010, pp. 122–136.

[139] ——, "Verifiability, privacy, and coercion-resistance: New insights from a case study," in *Security and Privacy (SP), 2011 IEEE Symposium on*, May 2011, pp. 538–553.

[140] A. Essex, J. Clark, and U. Hengartner, "Cobra: toward concurrent ballot authorization for internet voting," in *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, ser. EVT/WOTE'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=2372353.2372356

[141] S. Delaune, S. Kremer, and M. Ryan, "Coercion-resistance and receipt-freeness in electronic voting," in *Computer Security Foundations Workshop, 2006. 19th IEEE*. IEEE, 2006, pp. 12–pp.

[142] T. Moran and M. Naor, "Receipt-free universally-verifiable voting with everlasting privacy," in *Advances in Cryptology - CRYPTO 2006*, ser. Lecture Notes in Computer Science, C. Dwork, Ed. Springer Berlin / Heidelberg, 2006, vol. 4117, pp. 373–392, 10.1007/11818175_22. [Online]. Available: http://dx.doi.org/10.1007/11818175_22

[143] J. Benaloh, "Rethinking voter coercion: The realities imposed by technology," in *EVT/WOTE*. Berkeley, CA: USENIX, 2013, pp. 82–105. [Online]. Available: https://www.usenix.org/conference/evtwote13/workshop-program/presentation/Benaloh

[144] R. Abdelkader and M. Youssef, “Uvote: A ubiquitous e-voting system,” in *Mobile, Ubiquitous, and Intelligent Computing (MUSIC), 2012 Third FTRA International Conference on*, 2012, pp. 72–77.

[145] G. Tsoukalas, K. Papadimitriou, and P. Louridas, “From helios to zeus,” in *Greek Research and Education Network; Panayiotis Tsanakas, National Technical University of Athens, 2013 Usenix EVT conference*, 2013, pp. 1–10.

[146] B. Adida, “Helios: Web-based open-audit voting,” in *Proceedings of the 17th Conference on Security Symposium*, ser. SS’08, 2008, pp. 335–348.

[147] R. Haenni and O. Spycher, “Secure internet voting on limited devices with anonymized dsa public keys,” in *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, ser. EVT/WOTE’11, 2011.

[148] A. P. Adewole, A. S. Sodiya, and O. A. Arowolo, “A receipt-free multi-authority e-voting system,” *Int. Jour. of Comp. App.*, vol. 30, no. 6, pp. 15–23, Sep. 2011.

[149] H. Pan, E. Hou, and N. Ansari, “E-note: An e-voting system that ensures voter confidentiality and voting accuracy,” in *IEEE International Conference on Communications (ICC)*, June 2012, pp. 825–829.

[150] J. Bohli, J. Müller-Quade, and S. Röhrich, “Bingo voting: Secure and coercion-free voting using a trusted random number generator,” in *Proceedings of the 1st International Conference on E-voting and Identity*, ser. VOTE-ID’07, 2007, pp. 111–124.

[151] D. Sandler, K. Derr, and D. S. Wallach, “Votebox: A tamper-evident, verifiable electronic voting system,” in *Proceedings of the 17th Conference on Security Symposium*, ser. SS’08, 2008, pp. 349–364.

[152] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, “Prêt à voter: a voter-verifiable voting system,” *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 4, pp. 662–673, Dec 2009.

[153] A. Kiayias, M. Korman, and D. Walluck, “An internet voting system supporting user privacy,” in *Proceedings of the 22Nd Annual Computer Security Applications Conference*, ser. ACSAC’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 165–174.

VITA

VITA

Huian obtained his bachelor degree of computer science at Changsha Institute of Technology in China, and then the master degree of computer science at Institute of Computing Technology, Chinese Academy of Sciences in China. Later he enrolled in Department of Computer & Information Science at IUPUI and earned another master degree majoring in databases.

He has extensive experience in high performance computing. His research interests focus on data privacy and electronic voting.