Assurable, Transparent, and Mutual Restraining E-voting Involving Multiple Conflicting Parties

Xukai Zou*, Huian Li†, Yan Sui*, Wei Peng* and Feng Li‡
*Department of Computer and Information Science
Indiana University-Purdue University Indianapolis, Indiana 46202, USA
Email: {xkzou, huili, y sui, pengw}@cs.iupui.edu
†Department of Computer and Information Technology
Indiana University-Purdue University Indianapolis, Indiana 46202, USA
Email: fengli@iupui.edu

Abstract—E-voting techniques and systems have not been widely accepted and deployed by society due to various concerns and problems. One particular issue associated with many existing e-voting techniques is the lack of transparency, leading to the failure to deliver voter assurance. In this work, we propose an assurable, transparent, and mutual restraining e-voting protocol that exploits the existing two-party political dynamics in the US. The proposed 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, in combination, 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.

Index Terms—E-voting, Verifiability, Anonymity, Privacy, Assurance, Transparency, Mutual restraining voting

I. INTRODUCTION

Voting is the pillar of modern democracies, and a voting system is the central piece for people to execute their rights. Traditional voting, however, suffers from both low efficiency and unintentional errors. The event surrounding the 2000 US presidential election demonstrated the shortcomings of punch-cards and other antiquated voting systems. This drives the government to deploy more advanced voting systems.

Electronic-voting (e-voting) has been an active research topic with many advantages over traditional voting, but poses its own unique challenges. For example, if a discrepancy is found in tally, votes need to be recounted and the source of the discrepancy needs to be identified. The recounting and investigation 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 [16], [26].

Several e-voting solutions [39], [28], [37], [53] have been proposed. Some suggest keeping non-electronic parallels of electronic votes, or saving copies of votes in portable storage devices. These solutions either fail to identify sources of discrepancy or are susceptible to vote selling/coercion. Most solutions are based on cryptographic techniques, such as secret sharing, mix-net, and blind signature. These solutions are often opaque: Besides casting their votes, voters do not directly participate in collecting and tallying votes. This raises concerns over the trustworthiness and transparency of the voting process. In addition, these solutions sometimes entrust the fairness of the voting process to the impartiality of authorities. Voting under multiple conflicts-of-interest parties is not addressed by these solutions.

In this work, we propose a robust, assurable, transparent, and mutual restraining e-voting protocol that exploits conflicts of interest in multiple tallying authorities, such as the two-party political system in the US. The new protocol 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. These new techniques function as below:

- The tallied voting vector allows both individual and universal verification without jeopardizing voters’ privacy and anonymity. A voter is assured that his vote is counted, thus achieving voter assurance.
- Mutual lock voting ensures that a voter can cast one and only one vote, thus preventing multiple and fake voting.
- The in-process check and enforcement mechanism verifies any voter’s vote in an anonymous manner, identifies any invalid vote caused by misconduct, and ensures that voters follow the voting protocol without deviation. The trust is split equally among tallying authorities who have conflicting interests and will check and restrain each other.
- Both the vote-casting process using simplified \((N,N)\) secret sharing, and the vote-tallying process based on incremental aggregation (i.e., simple integer addition), are transparent (viewable) to voters. This, in turn, will further win trust and confidence of voters.

To the best of our knowledge, this is the first fully transparent e-voting protocol and is able to deliver voter assurance. The
The most groundbreaking feature of our e-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 aggregations, and 2) vote-tallying assurance – any voter is assured that their vote is viewably 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.

We understand that our protocol may not scale to very large number of voters. However, a typical voting system [57] has a hierarchical tree structure where vote totals from precincts (say, towns or counties) are sent to an upper level for consolidating. In such a voting structure, each precinct can apply our protocol on its own efficiently because of the relatively low population. Consequently, scalability is not a significant issue. However, we will address this in our future work, so that the protocol can apply to more general cases.

The rest of the paper is organized as follows. Related works are reviewed in Section II. Section III introduces models/assumptions and building blocks. The technical details of the protocol are presented in Section IV, followed by proof and analysis of the protocol in Section V. Simulation results are shown in Section VI. Section VII concludes our paper.

For convenience, notations used in the following sections are summarized in Table I for reference.

---

**TABLE I: Notations**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V, V_1, \ldots, V_N$</td>
<td>Voters, 1, \ldots, $N$ are voters’ indices</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of voters</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of candidates</td>
</tr>
<tr>
<td>$C, C_1, C_2$</td>
<td>Collectors</td>
</tr>
<tr>
<td>$Z_A, g$</td>
<td>Finite cyclic group, generator of the group</td>
</tr>
<tr>
<td>$v_i$</td>
<td>$V_i$’s voting vector</td>
</tr>
<tr>
<td>$v_i^f, v_i^b$</td>
<td>$V_i$’s forward and backward voting values</td>
</tr>
<tr>
<td>$V_A, L_A$</td>
<td>Aggregated voting vector, $V_i$’s location vector</td>
</tr>
<tr>
<td>$L_i$</td>
<td>$V_i$’s chosen location in one round</td>
</tr>
<tr>
<td>$L$</td>
<td>Length of location vector</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Length of each voter’s voting vector</td>
</tr>
<tr>
<td>$L_i^r$</td>
<td>Voter $V_i$’s real location (which is private to $V_i$) or to say, voter $V_i$’s row number</td>
</tr>
<tr>
<td>$L_i^{r_1}, L_i^{r_2}, \ldots, L_i^{r_E}$</td>
<td>$V_i$’s voting positions/bits or to say, the columns in row $L_i$</td>
</tr>
<tr>
<td>$v_i$</td>
<td>$V_i$’s voting vector</td>
</tr>
<tr>
<td>$v_i^f, v_i^b$</td>
<td>$V_i$’s forward and backward voting values</td>
</tr>
<tr>
<td>$v_i$</td>
<td>$V_i$’s voting vector</td>
</tr>
<tr>
<td>$v_i^f, v_i^b$</td>
<td>$V_i$’s forward and backward voting values</td>
</tr>
<tr>
<td>$v_i$</td>
<td>$V_i$’s voting vector</td>
</tr>
<tr>
<td>$v_i^f, v_i^b$</td>
<td>$V_i$’s forward and backward voting values</td>
</tr>
<tr>
<td>$v_i$</td>
<td>$V_i$’s voting vector</td>
</tr>
<tr>
<td>$v_i^f, v_i^b$</td>
<td>$V_i$’s forward and backward voting values</td>
</tr>
</tbody>
</table>

---

**II. RELATED WORKS**

The voting technology has experienced tremendous progress over the years. Extensive research on voting, particularly e-voting recently, has been conducted and different voting techniques and systems have been proposed in [6], [18], [19], [52], [55], [38], [54], [23], [34], [25], [39], [28], [37], [53], [30], [5].

Most voting techniques are based on cryptography, such as mix-nets, blind signature, homomorphic encryption, and secret sharing. The first voting scheme was proposed by Chaum [10] utilizing anonymous channels (so-called mix-nets) in 1981. Since then, more schemes and practical systems based on mix-nets have been proposed [15], [35], [56], [12], [31].

A blind signature allows an authority to sign an encrypted message without knowing the message’s context [42], [41], [33], [11], [9]. However, it is difficult to defend against misbehavior by the authority. In addition, some participants (e.g., the authority) know the intermediate results before the counting stage. This violates fairness of the voting process. 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. propose using a linkable ring signature, in which messages signed by the same member can be correlated, but not traced back to the member [17]. A scheme combining blind signature and mix-nets is proposed in [41].

E-voting schemes based on homomorphic encryption can trace back to the seminal works by Benaloh [3], [7] and later developments in efficiency [49], [20], and receipt-freeness [27], [36], [1], [45]. Rjaskova’s scheme [27], [36], [1] achieves receipt-freeness by using deniable encryption, which allows a voter to produce a fake receipt to confuse the coercer.

Several e-voting schemes exploit homomorphism that is provided by secret sharing [27], [36], [45], [3], [7], [1]. Some schemes [20], [49] are based on Shamir’s threshold secret sharing [51] and focus on providing universal verifiability, privacy, and robustness for e-voting. Iftene proposes a binary (yes/no) e-voting scheme based on the Chinese remainder theorem and oblivious transfer [29]: A voter sends shares of his vote to a set of subservers; each subserver sends aggregated shares to a central server for tallying. The scheme does not provide individual verifiability and accountability: A misbehaving voter can disrupt the entire process.

Experimental voting systems include Prêt à Voter [15], Scytl [50], ADDER [32], Helios [2], Punchscan/Scantegrity [24], [14], [13], ThreeBallot [43], [44], Bingo Voting [8], VoteBox [48], Prime III [21], SplitBallot [40], and STAR-Vote [5].

VoteBox [48] utilizes a distributed broadcast network and replicated log, providing robustness and auditability in case of failure, misconfiguration, or tampering. Prime III [21] is a multimodal voting system especially devoted to the disabled. SplitBallot [40] is a (physical) split ballot voting mechanism by splitting the trust between two conflict-of-interest parties/tallying authorities. Interestingly, ThreeBallot [44] is
a multi-ballot protocol that provides some of the benefits of a cryptographic voting system but without using cryptography. Scytl [50] uses a verification module (a physical device) on top of DRE. The trust, previously on the DRE, was transferred to the verification module. This solution assumes that the verification module is trusted, which may result in a “single point of failure”. STAR-Vote [5] is also a DRE-style system. Prêt à Voter [15] encodes a voter’s vote using a randomized candidate list. Vote privacy is ensured through randomization.

Unfortunately, due to the strict and conflicting requirements in voting, as indicated in [25], there is no current scheme/system satisfying all voting properties. For example, Grewal et al. [26] acknowledge that voter-coercion is hard to address so they redefine its meaning/scope. Helios [2], using Benaloh vote-casting approach [4] and the Sako-Kilian mixnet [46], has been proven to suffer from clash attacks [34].

### III. Assumptions and Building Blocks

Here we present security assumptions and building blocks of our protocol. Web based bulletin board is also briefly discussed.

#### A. Assumptions and Attack Models

Suppose there are \(N\) (\(N > 3\)) voters, \(V_i\) (\(1 \leq i \leq N\)), and two tallying parties, or collectors, \(C_1\) and \(C_2\). \(C_1\) and \(C_2\) have conflicting interests: Neither will share information with the other. The assumption of the existence of multiple conflict-of-interest collectors was previously proposed by Moran and Naor [40], and applied to real-world scenarios like the two-party political system in the US.

In our model, collectors mutually check and restrain each other, and thus, are assumed to follow the protocol. However, unlike many previous works, we do not assume they are fully trustworthy, but only that they will not collude with each other due to conflicts of interest. 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 the perpetrators without compromising honest voters’ privacy.

Although \((N,N)\) secret sharing theoretically involves mutual interaction among all voters, only a secure unicast channel between a voter and each collector is needed. Such a secure channel can be easily provided by equipping each collector with a public key cryptosystem.

We consider two types of adversaries: passive and active adversaries. Passive adversaries honestly follow the protocol but try to infer more information, while active adversaries (i.e., misbehaving voters) aim to violate the protocol by multiple voting, disturbing others’ voting, disturbing the tally and eventually bringing down the protocol. When we talk about disturbing others’ voting, we assume that there is no reason/incentive for a voter to give up his own voting right and disturb other unknown voters. However, he may disturb others and cast his vote simultaneously; this potential threat will be analyzed.

Based on the unconditional security of \((N,N)\) secret sharing (see Theorem 1 in [58]), a voter cannot infer any bit of information about any other voter’s vote from the shares and information given by other voters. Furthermore, even though \(k\) voters collude, as long as \(k \leq N - 2\), they together cannot gain any bit of information about the vote of any of the other voters either. Due to the in-process check and enforcement during the vote casting, misbehavior of any voter can be pinpointed during the vote casting and the dishonest voter can be identified.

#### B. Technical Components (TPs)

**TP1: Universal Verifiable Voting Vector.** For \(N\) voters and \(M\) candidates, a voting vector \(v_i\) 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 a row. Via a robust location anonymization scheme described in Section IV-B, each voter secretly picks a unique row. A voter \(V_i\) will put a 1 in the entry at the row and column corresponding to a candidate \(V_i\) votes for (let the position be \(L_i\)), and put 0 in all other entries. During tallying, all voting vectors will be aggregated. From the tallied voting vector (denoted as \(\text{V}_A\)), the votes for candidates can be incrementally tallied. Any voter can check his vote and also visually verify that his vote is indeed counted in the final tally. Furthermore, anyone can verify the vote totals for each candidate.

**TP2: 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 \(v_i = 2^{L_i - 1}\) and a backward value \(v'_i = 2^{L_i - 1}\) can be derived. Importantly, \(v_i \times v'_i = 2^{L_i - 1}\), regardless which candidate \(V_i\) votes for. During the vote-casting, \(V_i\) uses simplified \((N,N)\)-SS to cast their 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.

**Notes:** 1) 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 \(\text{V}_A\) and \(\text{V}'_A\) will be 0. If this happens, \(C_1\) and \(C_2\) can jointly find this location and then, along with the information collected during location anonymity, identify the perpetrator. 2) To prevent a collector from having all \(N - 1\) shares for a voter, each collector creates half of \(N - 1\) shares. 3) Interestingly, the new e-voting model deliberately distinguishes between a private vote and its secret ballot. Different from existing voting systems, a voter’s vote is kept secret to himself. However, its corresponding ballot, even called secret ballot, is revealed to the public in the vote-casting.

**TP3: In-process Check and Enforcement.** During the voting processes, collectors will jointly perform two cryptographic checks on the voting values of each voter. The first check uses STPM to prevent a voter from wrongly generating his share in

---

1The protocol can be extended to more than 2 with no essential difficulties.
the vote-casting stage. The second check prevents a voter from publishing an incorrect secret ballot when collectors collect it from a voter. The secret ballot is the modular addition of a voter’s own share and the share summations that the voter received from collectors.

C. Cryptographic Primitives

1) Simplified \((N,N)\) Secret Sharing \((\text{(N,N)}-\text{SS})\): A secret \(s\) is partitioned into \(N\) shares \(s_i\) \((1 \leq i \leq N)\) such that \(s = \sum_{i=1}^{N} s_i\). For a group of \(N\) members, each receives one of the shares. All \(N\) members need to pool their shares to recover \(s\). This secret sharing scheme is additively homomorphic [58]: The sum of two shares \(s_i\) and \(s_i'\) (corresponding to secrets \(s\) and \(s'\) respectively) is a share of the sum of the two secrets \(s\) and \(s'\).

2) Secure Two-party Multiplication (STPM): STPM is proposed by Samet and Miri [47]. Initially, each of parties, \(M_i\) \((i = 1, 2)\), 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 \(E(x_1, e) \times E(r_2, e)^{-1}\), and sends the result back to \(M_1\). 4) \(M_1\) decrypts the received value into \(D(\mu(x_1, e) \times E(r_2, e)^{-1}, d)\) and takes it as \(r_1\).

D. A Web Based Bulletin Board

A web based bulletin board allows anyone to monitor the dynamic vote casting and tallying in real time. This makes the voting and tallying process totally visible (i.e., transparent) to all voters. The bulletin board will dynamically display 1) ongoing 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 counts. 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 between vote-casting & ballot confirmation and vote-tallying & verification so that both voter privacy and voter assurance can be achieved simultaneously. This is a unique feature of the new e-voting system, comparing to all existing ones.

IV. MUTUAL RESTRAINING VOTING PROTOCOL

We first elaborate on each of the stages along with the applications of the technical components (TPs), and then give the design of our location anonymity scheme (LAS).

A. Description of Voting Stages with applications of TPs

Stage 1: Initialization. The following computations are carried out on a cyclic group \(\mathbb{Z}_{A_1}\), on which the Discrete Logarithmic Problem (DLP) is intractable. \(A = \max\{A_1, A_2\}\), in which \(A_1\) is a prime larger than \(2^{1024}\) and \(A_2\) is a prime larger than \(2^{2L} - 2^{L+1} + 1\). Let the number of voters be \(N\) and the number of candidates be \(M\). \(V_1, \ldots, V_N\) are voters.

Each voter generates his voting vector by the following two steps (TP1): 1) Voter \(V_i\), by collaboratively executing a LAS (Section IV-B) with other voters, obtains a unique and secret location \(L_i\). 2) \(V_i\) arranges a voting vector \(v_i\) of the length \(L = N \times M\) bits into \(N\) rows (corresponding to \(N\) voters) and \(M\) columns (corresponding to \(M\) candidates); \(V_i\) fills a 1 in his first row (i.e., the \(L_i\)th row) and the column for the candidate he votes, and 0 in all other entries. This arrangement 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 \(v_i\) (with a singleton 1 and all other entries 0), \(V_i\) derives two decimal numbers \(v_i\) and \(v_i'\) from \(v_i\): 1) \(v_i\) is the decimal number corresponding to the binary string represented by \(v_i\); 2) \(v_i'\) is the decimal number corresponding to \(v_i\) in reverse. In other words, if \(V_i\) sets the \(L_i\)th bit of \(v_i\) to 1, we have \(v_i = 2^{L_i-1}\) and \(v_i' = 2^{L_i-1}\), thus \(v_i \times v_i' = 2^{L_i-1}\); \(v_i\) and \(v_i'\) are said to be mutually retracted. This technique will be used to develop 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, \(V_i\) indirectly shares \(v_i\) and \(v_i'\) with other voters using \((N,N)\)-SS. Note that the sharing processes of \(v_i\) and \(v_i'\) are the same but independent from each other. Our initial design is to let each voter create \(N\) shares and then distribute \(N-1\) shares to the rest voters. However, this requires synchronous operations among all voters. An alternate design allows asynchronous operations, but still retains all properties of \((N,N)\)-SS.

In this asynchronous vote-casting, voters do not need to interact with each other. Instead, two collectors generate respective shares for all voters. They work with every voter with steps as below:

- Two collectors generate \(N-1\) shares for each voter in advance, half per voter per collector.
- Whenever a voter \(V_i\) logs into the system to cast vote, two collectors will each send their half shares (in fact, the sum of these shares) to this voter, i.e., \(C_j(j=1,2)\) sends \(S_{i,C_1}\) to \(V_i\). \(V_i\) computes his own share \(s_{i} = v_i - S_{i,C_1} - S_{i,C_2}\). Similarly, \(V_i\) gets \(s_{i}'\). \(V_i\) then sends his commitments (i.e., \(g^s_i, g^{s_i'}\)) to two collectors. Notes: under the assumption that two collectors have conflicting interests, neither of them should be able to derive \(V_i's\) vote from his commitment.
- Two collectors verify \(V_i's\) vote using Sub-protocol 1 (described later) and if passed, send the shares of the other \(N-1\) voters (one for each voter) to \(V_i\). Specially, \(C_j\) sends \(S_{i,C_j}\) (i.e., the sum of the shares \(C_j\) generates for the \(N/2\) voters) to \(V_i\). \(V_i\) then publishes the secret ballot \(p_i = s_{i} + S_{i,C_1} + S_{i,C_2}\). Two collectors can verify \(V_i's\)
obtain the candidate that are naturally supported by this protocol.

Stage 3: Collection/Tally. Collectors (and voters if they want) 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 $V_A$ of length $L$. The same is done for $p'_i$ ($1 \leq i \leq N$) to obtain $P'$, and consequently $V_A$. If voters have followed the protocol, these two vectors will 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 $V_A$ or $V_A'$ from being computed. Two possible solutions are: 1) after LAS, let all voters cast a default vote (e.g., abstain), and 2) subtract $S_i,c_1$ and $S_i,c_2$ of a voter who didn’t cast from the aggregation of the secret ballots $p_i$ and $S'_i,c_1$ and $S'_i,c_2$ from $p'_i$.

Stage 4: Verification. Anyone can verify whether $V_A$ is a reverse of $V_A'$ and whether each voter has cast one and only one vote. $V_i$ can verify the entry $L_i^j$ (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 $V_A$ and $V_A'$. In summary, both individual and universal verification are naturally supported by this protocol.

Stage 5: In-process check and enforcement. 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 attacker inserting multiple 1s in the voting vector. The following two sub-protocols, which are collectively known as in-process check and enforcement (TP3), ensure that each voter will put a single 1 in his voting vector, i.e., vote once and only once.

Sub-protocol 1: 1) Recall that in Stage 2, $C_j(j = 1, 2)$ has $S_i,c_j$ and $S'_i,c_j$ for $V_i$.

2) Since $V_i$ publishes $g^{s_i}$ and $g^{s'_i}$, $C_1$ can compute and publish $(g^{s_i})^{S_i,c_1}$ and $(g^{s'_i})^{S_i,c_1}$. In addition, $C_1$ publishes $g^{S_i,c_1}g^{S_i,c_1}$. Similarly, $C_2$ publishes $(g^{s_i})^{S_i,c_2}$, $(g^{s'_i})^{S_i,c_2}$, and $g^{S_i,c_2}g^{S_i,c_2}$.

3) $C_1$ and $C_2$ cooperatively compute $g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}$. A straightforward application of Diffie-Hellman key agreement [22] to obtain $g^{S_i,c_1}g^{S_i,c_1}$ and $g^{S_i,c_1}g^{S_i,c_1}$ will not work.2 Therefore, STPM is used to compute $g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}$ without disclosing $g^{S_i,c_1}, g^{S_i,c_1}, g^{S_i,c_1}$ and $g^{S_i,c_1}$ as follows:

- Executing STPM, $C_1$ and $C_2$ obtain $r_1$ and $r_2$ respectively such that $r_1 + r_2 = S_i,c_1$.
- Executing STPM, $C_1$ and $C_2$ obtain $r'_1$ and $r'_2$ respectively such that $r'_1 + r'_2 = S_i,c_1$.
- $C_1$ computes $g^{r_1 + r'_1}$ and $C_2$ computes $g^{r_2 + r'_2}$ and then they exchange the results.
- Both collectors obtain $g^{r_1 + r'_1 + r'_1 + r_2} = g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}g^{S_i,c_1}$.

4) Each collector uses $V_i$’s commitment and the above computation results to obtain $g^{w_1} \times (g^{w_1})^{S_i,c_1} \times (g^{w_1})^{S_i,c_1} \times g^{S_i,c_1}g^{S_i,c_1} \times (g^{w_1})^{S_i,c_1} \times (g^{w_1})^{S_i,c_1} \times g^{S_i,c_1}g^{S_i,c_1} \times g^{S_i,c_1}g^{S_i,c_1}$. The collectors can verify that the product equals $g^{2^{L-1}}$. If not, $V_i$ must have shared $v_j$ or $v'_j$ incorrectly.

Sub-protocol 2: While Sub-protocol 1 ensures that $V_i$ generates $s_i$ properly, Sub-protocol 2 enforces that $V_i$ will faithfully publish the secret ballots, $p_i$ and $p'_i$.

1) Recall that $C_j(j = 1, 2)$ has $S_i,c_1$ and $S'_i,c_1$ for $V_i$, so $C_j$ publishes $g^{S_i,c_1}$ and $g^{S'_i,c_1}$.

2) From the published $p_i$ and $p'_i$, the collectors compute $g^{p_i}$ and $g^{p'_i}$. Since $g^{s_i}$ and $g^{s'_i}$ are published and verified in Sub-protocol 1, collectors will verify that $g^{s_i}g^{S_i,c_1}g^{S_i,c_1} = g^{p_i}$ and $g^{s'_i}g^{S_i,c_1}g^{S_i,c_1} = g^{p'_i}$. If either of these fails, $V_i$ must have published the wrong secret ballots $p_i$ and/or $p'_i$.

B. Design of a Robust and Efficient LAS

Inspired by the work in [58], we propose a new location anonymity scheme (LAS) that is robust and efficient. Our new scheme solves the following problem with the previous schemes: 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 [58] may never finish. Our new LAS is based on the mutual lock voting mechanism and works as follows:

1) Each voter $V_i$ initializes a location vector $L_i$ (of length $L$) with 0s. $V_i$ randomly selects a location $L_i$ ($1 \leq L_i \leq L$) and sets the $L_i$th element/bit of $L_i$ to 1.

2) From $L_i$, $V_i$ obtains two values $l_i$ and $l'_i$ by: 1) encoding $L_i$ into a decimal number $l_i$; and 2) reversing $L_i$ to be $l'_i$ and encoding it into a decimal number $l'_i$. For example, if $L_i = [000010]$, we obtain $l_i = 10$ and $l'_i = 10000$. Evidently, $l_i \times l'_i = 10^{L_i-1}$.

3) $V_i$ shares $l_i$ and $l'_i$ using $(N,N)$-SS as in Stage 2. All voters can obtain the aggregated location vector $L_{A}$ and $L_{A}$. If $V_i$ has followed the protocol, $L_{A}$ and $L_{A}$ are the reverse of the other.

A decimal encoding, instead of a binary one, is used to encode $L_i$. 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 $L_i = 000010$, $L_j = 000010$, and $L_k = 000100$, respectively. Therefore, $L_{A} = 001000$: Voters cannot tell if they have obtained unique locations. This will not be the case if $L_i$ uses a larger base; however, encoding $L_i$ 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 \( L_i \)th element/bit of \( L_s \) is 1. If so, \( V_i \) has successfully selected a location without colliding with others. \( V_i \) also checks if everyone has picked a location without colliding with others by examining whether \( \max(L_s) = 1 \). 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 select the same location, while others randomly select from locations not been chosen.

5) The in-process check and enforcement mechanism (Stage 5) is concurrently executed by collectors to enforce that a voter will select one and only one location in each round. Furthermore, the mechanism (proved in Section V-B) ensures that any attempt of inducing collision by deliberately selecting an occupied position will be detected and, 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 \( L_v = \sum_{j=1}^{L_s} (L_s)_j \). After the adjustment, the occupied locations become contiguous. The length of the adjusted \( L_v \), denoted as \( L \), equals to the number of voters, \( N \).

We will complement the above discussion with analysis (Section V-B) and simulation results (Section VI).

**Notes:** 1) Location anonymity, a special component in our protocol, 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 anonymity can be executed asynchronously. 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 that neither can determine which voter gets which number, even though a collector may collude with some voter(s).

**V. Property Proof and Analysis**

**A. Proof of Robustness of Voting**

The protocol is robust in the sense that a misbehaving voter will be identified. A misbehaving voter \( V_i \) may:

- submit an invalid voting vector \( V_i \) (\( V_i' \)) with more than one (or no) 1s;
- generate wrong \( s_i \) (\( s_i' \)), thus wrong commitment \( g^{s_i} \) (\( g^{s_i'} \));
- publish an incorrect secret ballot \( p_i \) (\( p_i' \)) such that \( p_i \neq s_i \) and \( s_i + s_i' + L_c \), \( p_i' \neq s_i + L_c \).

First, we show that a voter submitting an invalid voting vector \( V_i \) (\( V_i' \)) with more than one 1s will be detected. Without loss of generality, we assume two positions, \( L_i \) and \( L_i' \), are set to 1. (A voter can also misbehave by putting 1s in 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

\[
V_i = 2^{(L_i - L_i)} + 2^{(L_i - L_i')}, V_i' = 2^{(L_i' - 1)} + 2^{(L_i' - 1)},
\]

\[
V_i \times V_i' = 2^{L_i - 1} + 2^{L_i - 1} + 2^{L_i - L_i'} + 2^{L_i - L_i'} - 1 + 2^{L_i' - L_i} + 2^{L_i' - L_i} - 1.
\]

All the computations are moduli operations. If we use \( Z_n \), which has at least \( 2^{L_i + L_i - 1} + 1 \) elements/bits, we have \( V_i \times V_i' \neq 2^{L_i - 1} \) and \( g^{V_i} \neq g^{V_i'} \). Assuming \( V_i \) generates an invalid voting vector without being detected, this will lead to the following contradiction by Sub-protocol 1:

\[
g^{2^{L_i - 1}} = g^{(V_i + V_i')} = g^{(s_i + s_i'))} \times g^{(s_i + s_i')}, g^{s_i + s_i'} = g^{2^{L_i - 1}}.
\]

Since the computation is on \( Z_n \), we have: \((s_i + s_i') (s_i + s_i') = 2^{L_i - 1}\). Given that:

\[
s_i + s_i + s_i' \neq V_i, s_i' + s_i + s_i' \neq V_i',
\]

there must exist one and only one position \( L_i \) which is set to 1 and \( L_i' \neq L_i \). This indicates that \( V_i \) gives up his own voting positions, but votes at a position assigned to another voter. In this case, \( V_i \)'s voting positions in \( V_A \) and \( V_A' \) will be 0. 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 a \( N \times M \) array).

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 \( s_i + s_i + s_i + s_i' \neq p_i \), we obtain \( g^{s_i + s_i + s_i + s_i'} \neq g^{p_i} \). This indicates that \( V_i \) gives up his own voting positions, but votes at a position assigned to another voter. In this case, \( V_i \)'s voting positions in \( V_A \) and \( V_A' \) will be 0. 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 a \( N \times M \) array).

**B. Proof of Robustness of Location Anonymity**

The analysis in Section V-A 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.

---

*Unless, of course, another voter puts a 1 in \( V_i \)'s position. We can either trace this back to a voter that has its positions all 0s, or there is a loop in this misbehaving chain, which causes no harm to non-misbehaving voters.*
Let the collision happen at $L_i$, i.e., $L_i$ is chosen by $V_i$ in the previous round, and both $V_i$ and $V_j$ claim $L_i$ in the current round. In this case, $V_j$ is the voter who deliberately induces collision. To identify a voter who chooses $L_i$ in a given round, $C_1$ and $C_2$ do the following collaboratively. For each voter, using the STPM, $C_1$ and $C_2$ compute $Q = g^{s_i C_1 + s_i C_2} \left( Q = g^{s_i C_1 + s_i C_2} \right)$ and check whether $g^{s_i O_i} / g^{s_i} = Q \left( g^{s_i O_i} / g^{s_i} = Q \right)$. By doing this, the collectors identify the voter who selects $L_i$ without divulging other voters’ locations. Although the honest voter $V_i$ who chooses $L_i$ is exposed along with the malicious $V_j$, $V_i$ can restore location anonymity by selecting another position in the next round and $V_j$ should be punished.

C. Analysis of Main Properties

**Correctness.** If all voters are honest and correctly follow the protocol, $V_A \left( V_A \right)$ is the aggregation of all votes. Assume that $N$ voters vote at positions $L_1, L_2, \cdots, L_N$ ($L_i \neq L_j$) respectively. Each $V_i$ computes $v_i$ as $v_i = 2^{\left( L_i - L_1^i \right)} (i = 1, \cdots, N)$. Due to the homomorphism of $(N, N)$-SS, we have:

$$P = v_1 + v_2 + \cdots + v_N$$

$$= 2^{\left( L_i - L_1^i \right)} + 2^{\left( L_i - L_2^i \right)} + \cdots + 2^{\left( L_i - L_N^i \right)}$$

$$V_A = v_1 + v_2 + \cdots + v_N.$$

Since each voter votes at one of his own voting positions (i.e., $L_i \neq L_j$ where $i \neq j$), there is no carry in the additions. Thus, each $V_i$ can check the $L_i$th bit of $V_A$ to see if his vote has been correctly counted. Similar arguments apply to $V_A$.

**Anonymity.** The protocol preserves anonymity if no more than $N - 2$ voters collude. The claim follows the proof in [58]. Furthermore, the protocol splits trust, traditionally vested in a central authority, now between two non-colluding collectors with conflicts of interest.

**Verifiability.** Both individual and universal verifiability are achieved since anyone can verify if his vote and vote totals.

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

**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).

**Fairness.** Fairness is ensured due to the following 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 is the sum of all secret votes obtained in its entirety and in an anonymous manner. It is this sudden transition that precludes any preannunciation of partial voting results.

**Transparency and voter assurance.** The protocol is transparent in that voters participate in the whole voting process, rather than entrusting the process to a central authority like in many previous e-voting solutions.

D. Analysis of Protocol Performance and Complexity

In this section, we analyze the computational complexity of forward and backward mutual lock voting and in-process enforcement. Suppose that each message takes $T$ bits. Since the protocol works on a cyclic group $Z_A (\lambda = \max \{ A_1, A_2 \}$, in which $A_1$ is a prime greater than $2^{1024}$ and $A_2$ is a prime greater than $2^M - 2^{k+1} + 1$, we see that $T = O(L)$.

The forward and backward mutual lock voting involves two independent sharing processes of $v_i$ and $v_j$. A voter’s communication cost includes publishing his commitments and secret ballot $p_i$, so the total is $O(T)$. His computational cost includes computing $v_i, s_{i1}, p_i$, and the commitments (e.g., $g^{s_i}$), each costing $O(T)$, $O(T)$, $O(T)$ and $O(T^3)$ respectively. The same applies to the sharing of $v_j$. **Notes:** The commitments can be typically 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) sending sums of shares to each voter; 2) publishing $g^{s_{i1} C_{j1}}$, $g^{s_{i1} C_{j1}}$, and $g^{s_{i2} C_{j2}}$ for each voter; 3) publishing $g^{s_{i1} C_{j1} C_{j2}}$ and $g^{s_{i1} C_{j1} C_{j2}}$ for each collector; and 4) computing $g^{s_{i1} C_{j1}}$ or $g^{s_{i1} C_{j2}}$ for each voter.

Assume that the STPM messages are encoded into $\tilde{T}$-bits when computing $g^{s_{i1} C_{j1} C_{j2}}$. The communication costs are $O(T)$, $O(T)$, $O(\tilde{T})$, and $O(T)$, respectively. For $N$ voters, the total cost for each collector is $(O(T) + O(T) + O(T) + O(T)) N$.

The computation by $C_j$ involves: 1) generating half of $N - 1$ shares for each voter, which costs $O(N^2 T)$ totally; 2) deriving $S_{i1} C_{j1}$, $S_{i1} C_{j1}$, $S_{i1} C_{j1}$, and $S_{i1} C_{j1}$ during the vote-casting process which costs $O(N^2 T)$; 3) summing up the $p_i$ during voting collection/total which costs $O(N T)$; 4) the computational costs of Sub-protocol 1 and Sub-protocol 2. In Sub-protocol 1, both collectors: 1) compute $g^{s_{i1} S_{i1} C_{j1}} g^{s_{i1} S_{i1} C_{j1}}$ and $g^{s_{i2} C_{j1} C_{j2}}$ (for each voter) (j = 1, 2); 2) compute $g^{s_{i1} C_{j1} C_{j2}}$ and $g^{s_{i1} C_{j2} C_{j1}}$, which involve STPM; and 3) multiply the commitments. Computing $g^{s_{i1} C_{j1}} g^{s_{i1} C_{j1}}$, and $g^{s_{i1} C_{j1} C_{j2}}$ costs $O(T)$. Multiplying the commitments costs $O(T^2)$. Computing $g^{S_{i1} C_{j1} C_{j2}} \times S_{i1} C_{j2} C_{j1}$ consists of obtaining $r_1$ ($r_1'$) and $r_2$ ($r_2'$) with STPM, computing $g^{r_1 + r_2}$ and $g^{r_1'' + r_2''}$, and multiplying $g^{r_1 + r_2}$ and $g^{r_1'' + r_2''}$. Let the complexity for STPM be $O(TPMC)$. The cost of computing $g^{S_{i1} C_{j1} C_{j2}} \times S_{i1} C_{j2} C_{j1}$ is $O(TPMC) + O(T^3) + O(T^3)$. The total computational cost of Sub-protocol 1 is $O(T^3) + O(T^2) + O(TPMC)$ for each voter. In Sub-protocol 2, the collectors: 1) compute $S_{i1} C_{j1}$ and $S_{i1} C_{j2}$; 2) compute $g^{S_{i1} C_{j1}}$ and $g^{S_{i1} C_{j2}}$; 3) multiply $g^{p_i}$, $g^{S_{i1} C_{j1}}$ and $g^{S_{i1} C_{j2}}$, and also $g^{p_i}$, $g^{S_{i1} C_{j1}}$ and $g^{S_{i1} C_{j2}}$; and 4) compute $g^{p_i}$ and $g^{p_i}$. These computations cost $O(N T), O(T^3), O(T^3)$ and $O(T^3)$, respectively. The total cost of Sub-protocol 2 is $O(N T) + O(T^3) + O(T^2) + O(T^3)$ for each voter.

VI. Simulation Results

We implemented our protocol simulation in Java, and present results here. The experiments were carried out on a computer with a 1.87GHz CPU and 32G of memory. For each experiment, we took the average of 10 rounds of simulations.
1-out-of-2 voting process is simulated. Thus, the length of the voting vector is \( L = 2N \) where \( N \) is the number of voters.

![Fig. 1: Collectors run Sub-protocol 1 in TP3 against one voter](image1.png)

![Fig. 2: Collectors run Sub-protocol 2 in TP3 against one voter](image2.png)

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.

Figure 1 and 2 show the computation time of in Sub-protocol 1 and Sub-protocol 2 respectively. Sub-protocol 1 was dominated by STPM, due to the computationally intensive Paillier Cryptosystem used in our implementation. However, this should not be an issue in real life since the collectors usually possess greater computing power.

Figure 3 shows the time for one collector to collect and tally votes. The execution time depends on the number of voters \( N \) and the length of \( p_i = L \). As \( L \) increases, the voting collection/tally time increases by \( NL = O(L^2) \).

The simulation results confirm the performance analysis in Section V-D. 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.

![Fig. 3: One collector collects/tally votes](image3.png)

**VII. CONCLUSION**

We proposed a robust, assurable, transparent, and mutual restraining e-voting protocol that is designed to exploit the conflicts of interest in multiple tallying authorities, such as in the two-party political system in the United States. Our protocol is built upon three novel technical contributions—verifiable voting vector, forward and backward mutual lock voting, and proven in-process 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 voter assurance. Each voter can be assured that his vote is counted both technologically and visibly. Through the analysis and simulation, we demonstrated the robustness, effectiveness, and feasibility of our protocol.

We plan to further improve our protocol in the future, particularly, in two aspects. One is the scalability, and the other is vote selling and voter coercion.

**REFERENCES**


A. Kiayias, M. Korman, and D. Walluck. An internet voting system.


