cryptography

Article

Transparent, Auditable, and Stepwise Verifiable Online E-Voting Enabling an Open and Fair Election

Xukai Zou [1,*], Huian Li [1], Feng Li [2], Wei Peng [1] and Yan Sui [1]

1 Computer and Information Science Department, School of Science, Purdue University Indianapolis, Indianapolis, IN 46202, USA; huili@iu.edu (H.L.); 4pengw@gmail.com (W.P.); charlotte.yansui@gmail.com (Y.S.)
2 Computer and Information Technology Department, School of Engineering, Purdue University Indianapolis, Indianapolis, IN 46202, USA; fengli@iupui.edu
* Correspondence: xkzou@cs.iupui.edu; Tel.: +1-317-278-8576

Received: 24 May 2017; Accepted: 15 August 2017; Published: 17 August 2017

Abstract: Many e-voting techniques have been proposed but not widely used in reality. One of the problems associated with most 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, stepwise verifiable, viewable, 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/viewable to anyone, which allow any voter to visually verify that his vote is indeed counted and also allow any third party to audit the tally, thus, enabling open and fair election. Depending on the voting environment, our interactive protocol is suitable for small groups where interaction is encouraged, while the non-interactive protocol allows large groups to vote without interaction.

Keywords: e-voting; verifiability; anonymity; privacy; assurance; transparency; auditability

1. Introduction

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. One potential solution is to allow voters to cast ballots on Internet-enabled mobile devices [4].

Online voting (electronic voting) 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 cryptographic techniques, such as secret sharing, mix-nets, 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 conflicts-of-interest 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 (e.g., dropping the ballot into a ballot box), or by the opaque transition (e.g., due to encryption) from the vote-casting to the vote-tallying, thus damaging voter assurance associated with an open question: "Will my vote count?" [17].

In terms of opaqueness, the term was raised as early as in a 2004 paper [18]: "Voting machines are *black boxes* whose workings are opaque to the public." Later in 2008, the authors in [12] states "some people believe that any use of cryptography in a voting system makes the system too opaque for the general public to accept." Furthermore, authors in [19] states "once the vote is cast the voter *loses sight* of it." Finally, the authors in [20] states that the mixing part in Helios (and Zeus) is a black box to voters. Thus, in this paper, by opaqueness we mean that the use of complex cryptographic techniques involving encryption, mix-net, zero-knowledge, etc. causes some part of vote-casting, tallying and/or verification not viewable to the general public even though the background cryptographic techniques are mathematically sound and can fundamentally guarantee the integrity of the voting process and the tallying result. On the other hand, by transparency, we mean that the voting process from vote-casting, tallying, and verification is viewable to the general public and the integrity of the voting process and tally are visually verifiable by any one, besides technical guarantee of the integrity of the tallying result.

In this work, we propose a transparent, auditable, and step-wise verifiable voting protocol that exploits conflicts of interest 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 any one (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/visual tallying and verification. The cast ballots, sum(s) of ballots, votes, and sum(s) of votes (for each candidate) are all displayed on a public bulletin board. A voter or any one can view them, verify them visually, and even conduct summations of ballots (and as well as votes) himself. Thus, the newly proposed protocol delivers both individual verification and universal verification.

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 [21,22]. 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 it delivers visible 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 his 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, step-wise 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 broader voter assurance.

Real time check and verification of any cast secret ballots and their incremental aggregation, along with visibility and transparency, provide strong integrity and auditability of the voting process and final tally. In particular, we relaxed the trust assumption a bit in our protocol. Instead of assuming that tallying authorities who conduct tally are trustworthy, (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) the new protocol, like the split trust in the split paper ballot voting [23], 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, any attacks, regardless of from inside such as collusion among voters or between voters and one tallying authority 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. 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 paper. We assume collectors will not collude since they represent parties with conflicting interests.

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 [24] present a hierarchical structure such as a tree where voting 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 transmitted to an upper level central location for consolidation. We will have a more detailed discussion about this later.

The rest of the paper is organized as follows. Section 2 gives an overview of the protocol including assumptions and attack models. Building blocks are also introduced here. The technical details of the voting protocol are presented in Section 3, followed by security and property analysis in Section 4. Complexity analysis and simulation result are given in Section 5. Section 6 is the related work and protocol comparison. Section 7 discusses scalability of our protocol. Section 8 concludes our paper and lays out the future work.

This paper is an extension of our INFOCOM'14 work [25]. Mainly, we add the interactive protocol design in Section 3.1, modify **Sub-protocol 1** which suffered a possible brute-force attack originally in [25] in Section 3.2.1, along with the rigorous proof of the revised **Sub-protocol 1** in Section 4.1, and provide an example of bulletin board dynamics in Section 3.4. The attack model covers more diverse scenarios as mentioned in Section 2.1, with corresponding security analysis later in Section 4. Additional experiments were conducted and the results are presented in Section 5. The related work in Section 6 is more comprehensive. Section 7 is new for further discussion.

Notations used are summarized in Table 1.

Table 1. Notations.

(n,n)-SS	(n,n)-secret sharing
LAS	Location Anonymization Scheme
STPM	Secure Two-Party Multiplication
$V_1, \ldots, V_N$	Voters. $1, \ldots, N$ are voters' indices
N	Number of voters
M	Number of candidates
C_1, C_2	Collectors
$\mathbf{Z}^*_{\mathbf{A}}, g$	Finite cyclic group, and generator of the group
$\mathbf{v_i}$	V_i's voting vector
$v_i; v'_i$	V_i's forward and backward voting values
$\mathbf{V_A}; \mathbf{V'_A}$	Aggregated voting vector, and its reverse
$\mathbf{L_i}$	V_i's location vector
$\mathbf{L_A}$	Aggregated location vector
$\hat{L_i}$	V_i's Chosen location in one round
$\bar{L}$	Length of location vector
L	Length of each voter's voting vector
L_i	Voter V_i's real location (which is private to V_i) or to say, voter V_i's row number
$L^i_B, L^i_{B+1}, \ldots, L^i_E$	V_i's voting positions/bits or to say, the columns in row L_i
L^i_c $(L^i_B \leq L^i_c \leq L^i_E)$	A voting position where V_i sets to 1 (cast vote) also referred to as a voting element or bit
$s_{ij}(s'_{ij})$	V_i's share of v_i (v'_i)
$p_i; p'_i$	V_i's secret ballots for v_i and v'_i, respectively
$P; P'$	Sum of p_i, and sum of p'_i $(1 \leq i \leq N)$
$S_{i,C_j}; S'_{i,C_j}$	Sum of V_i's shares that C_j (j = 1, 2) creates in the interactive protocol or receives in the non-interactive protocol
$\tilde{S}_{i,C_j}; \tilde{S}'_{i,C_j}$	Sum of shares sent to V_i from other voters in the interactive protocol or C_j sends in the non-interactive protocol
$\mathbf{L_i}$	V_i's location vector (used in LAS)
$\mathbf{L_A}$	Aggregated location vector (used in LAS)
$\hat{L_i}$	V_i's chosen location in one round (used in LAS)
$\bar{L}$	Length of location vector (used in LAS)

2. Mutual Restraining Voting Overview

2.1. 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 (The protocol can be extended to more than two with no essential difficulties). The number of candidates in the election is M and each voter votes one out of M candidates. C_1 and C_2 *have conflicting interests*: Neither will share information (i.e., the respective shares they have about a voter's vote) with the other. The assumption of multiple conflict-of-interest collectors was previously proposed by Moran and Naor [23], and applies to real world scenarios like the two-party political system in the US.

For the interactive voting protocol presented in Section 3.1, *secure uncast channel* is needed between voters and between a voter and each collector. However, for the non-interactive protocol in Section 3.3, *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. Our protocols utilize two cryptographic primitives (1) simplified (n, n)-secret sharing and (2) tailored secure two-party multiplication (STPM) as building blocks. We also use DLP under its hardness assumption. Simplified (n, n)-secret sharing is used in vote-casting and STPM and DLP are used by two collectors to check the validity of the vote cast by the voter.

In terms of attack model, we assume that majority of voters are benign since the colluding majority can easily control the election result otherwise. The malicious voters can send inconsistent information to different parties or deliberately deviate from the protocol, e.g., trying to vote for more than one candidate. We will show that the protocol can detect such misbehaviors and identify them without compromising honest voters' privacy.

One fundamental assumption of the protocol is that the collectors are of conflict-of-interest and will not collude. Here "collusion" specifically means that the collectors exchange the shares of the voter's vote directly, thus, discovering the voter's vote (each of the two collectors has roughly half amount of shares respectively). However, they DO collaborate to check and enforce the voter to follow the protocol using STPM without compromising voter secrecy. Moreover, the collectors mutually check and restrain each other, and thus, are assumed to follow the protocol. However, unlike many protocols in which tallying authorities perform vote-tallying and certain trust is assumed on tallying authorities for the integrity of the tallying result (In some protocols, trustworthiness of tallying authorities may be achieved via cryptographic techniques such as commitment/zero knowledge proof), we do not assume trustworthiness on the collectors for such integrity in the new protocol. The primary role of the collectors in the protocol is to check the validity of the voter's vote, i.e., the vote is for one and only one candidate. It is not needed for the collectors to provide/guarantee vote secrecy or the integrity of tallying result. Tallying and verification can be conducted by voters themselves. In fact, if we assume voters are honest, they can execute the protocol themselves (that is why our interactive voting protocol works) without the need/involvement of collectors. If one collector is honest, the misbehaviors of other collectors will be detected. Furthermore, even though all collectors misbehave (independently), such misbehaviors will be easily detected with overwhelming probability because such behaviors will mostly result in an invalid voting vector.

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 paper, we consider the attacks targeting at the voting protocol 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 Section 4, all these attacks can either be prevented or detected.

2.2. Cryptographic Primitives

2.2.1. Simplified (n, n)-Secret Sharing ((n, n)-SS)

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 [26]. The scheme is additively homomorphic [27]: The sum of two shares $s_i + s'_i$ (corresponding to s and s', respectively) is a share of the secret $s + s'$.

Theorem 1. *The simplified (n, n)-SS scheme is unconditionally indistinguishable. That is, collusion of even up to $n - 2$ participants cannot gain any bit of information on the shares of the rest. The proof is given in [27].*

2.2.2. Tailored Secure Two-Party Multiplication (STPM)

Tailored STPM (Secure two-party multiplication (STPM) in the original paper [28] 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) is proposed in [28]. 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 decrypts the received value into $D(E(x_1, e)^{x_2} E(r_2, e)^{-1}, d)$, resulting in r_1.

2.3. *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 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.

2.4. *Technical Components (TPs)*

The protocol includes three important technical components.

TP1: 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 Section 3.5, 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.

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 (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.

TP3: 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 3.1 and 3.3.

2.5. High-Level Description of the E-Voting 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 Section 3.5.
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 **TP1** in Section 2.4.
   (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 **TP2** in Section 2.4.
   (c) Think of shares of all voters' ballots forming an $N \times N$ matrix (as shown in Table 2). 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 $s_{ii} = v_i - S_{i,C_1} - S_{i,C_1}$, 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 the first half of the voters (the "lower half"). Similarly, C_2 sends voter V_i the sum $\tilde{S}_{i,C_2}$ of the shares C_2 generated for the second half of the voters (the "upper half"). In the interactive protocol, V_i receives them 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 retains from this is s'_{ii} and the ballot from this is called p'_i and is also public.
   (e) Simultaneously with the previous step, the voter also publishes his commitments $(g^{s_{ii}}, g^{s'_{ii}}, g^{s_{ii}s'_{ii}})$, where g is the base for the discrete logarithm problem. Two authorities jointly verify the validity of each cast ballot. See **TP3** in Section 2.4.
3. Tally and verification. The authorities (in fact any one 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 (authority, 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 $\mathbf{V'_A}$, sum of each candidate's votes, and final tally.

Table 2. The collectors generate all shares for each voter, with each collector generating half of $N-1$ shares. v_i is V_i's vote. $s_{\hat{i,j}}$ ($i \neq j$) 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 ($s_{\hat{i,j}}$) 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 ($s_{\hat{j,i}}$) 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.

		$\leftarrow -$		C_1	$- \rightarrow$	$\leftarrow -$	C_2	$- \rightarrow$	
				$\tilde{S}_{i,C_1}$			$\tilde{S}_{i,C_2}$		
				$\Downarrow$			$\Downarrow$		
	v_1	$s_{1,1}$	$s_{\hat{1,2}}$	$\cdots$	$s_{\widehat{1,N/2}}$	$s_{\check{1,N/2+1}}$	$\cdots$	$s_{\check{1,N}}$	
$C_1, S_{i,C_1} \Rightarrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\Leftarrow S_{i,C_2}, C_2$
	$v_{N/2}$	$s_{\widehat{N/2,1}}$	$s_{\widehat{N/2,2}}$	$\cdots$	$s_{N/2,N/2}$	$s_{\check{N/2,N/2+1}}$	$\cdots$	$s_{\check{N/2,N}}$	
	$v_{N/2+1}$	$s_{\check{N/2+1,1}}$	$s_{\check{N/2+1,2}}$	$\cdots$	$s_{\check{N/2+1,N/2}}$	$s_{N/2+1,N/2+1}$	$\cdots$	$s_{\widehat{N/2+1,N}}$	
$C_2, S_{i,C_2} \Rightarrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\Leftarrow S_{i,C_1}, C_1$
	v_N	$s_{\check{N,1}}$	$s_{\check{N,2}}$	$\cdots$	$s_{\check{N,N/2}}$	$s_{\widehat{N,N/2+1}}$	$\cdots$	$s_{N,N}$	
				$\Uparrow$			$\Uparrow$		
				$\tilde{S}_{i,C_2}$			$\tilde{S}_{i,C_1}$		
		$\leftarrow -$		C_2	$- \rightarrow$	$\leftarrow -$	C_1	$- \rightarrow$	

3. Mutual Restraining Voting Protocol

In this section, we first 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 given next. In the end, we discuss the design of our location anonymization scheme.

3.1. 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 $2^{2L} - 2^{L+1} + 1$.

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 LAS (in Section 3.5) 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 (**TP1**). 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^i_cth bit of $\mathbf{v}_i$ to 1, we have $v_i = 2^{L-L^i_c}$ and $v'_i = 2^{L^i_c - 1}$, thus $v_i \times v'_i = 2^{L-1}$. v_i and v'_i are said to be *mutually restrained* (**TP2**). 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}. 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. The 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 3.2).

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 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 $\mathbf{V_A}$ or $\mathbf{V'_A}$ from being computed. If V_i's vote 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_c^i (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.

3.2. 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* (**TP3**), ensure that each voter should put a single 1 in his voting vector, i.e., vote once and only once.

3.2.1. Revised Sub-Protocol 1

Sub-protocol 1 in [25] suffers from a possible brute-force attack, so Step 3 here has been redesigned accordingly to avoid such an attack.

(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 [29] to obtain $g^{S_{i,C_1}S'_{i,C_2}}$ and $g^{S'_{i,C_1}S_{i,C_2}}$ will not work. (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 both vote anonymity (the vote is known) and voter privacy (the location is known)) 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 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 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. (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.)
- C_1 computes $g^{r_1+r'_1}$, C_2 computes $g^{r_2+r'_2}$. Obviously $g^{r_1+r'_1} \times g^{r_2+r'_2} = g^{r_1+r'_2+r'_1+r_2} = g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$. ($C_1$ 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 4.1.)

(4) C_1 computes a combined product $P_1 = (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times g^{r_1+r'_1}$ and similarly, C_2 computes a combined product $P_2 = (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{r_2+r'_2}$ and then they exchange P_1 and P_2.

(5) Using V_i's commitment $g^{s_{ii}s'_{ii}}$ and P_1, P_2, each collector obtain $g^{s_{ii}s'_{ii}} \times P_1 \times P_2 = g^{s_{ii}s'_{ii}} \times (g^{s_{ii}})^{S'_{i,C_1}} \times (g^{s'_{ii}})^{S_{i,C_1}} \times g^{S_{i,C_1}S'_{i,C_1}} \times (g^{s_{ii}})^{S'_{i,C_2}} \times (g^{s'_{ii}})^{S_{i,C_2}} \times g^{S_{i,C_2}S'_{i,C_2}} \times g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}}$. 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.

3.2.2. Sub-Protocol 2

While the revised 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 publish $g^{\tilde{S}_{i,C_j}}$. Similarly for v'_i, $C_j (j = 1, 2)$ will publish $g^{\tilde{S}'_{i,C_j}}$.

(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.

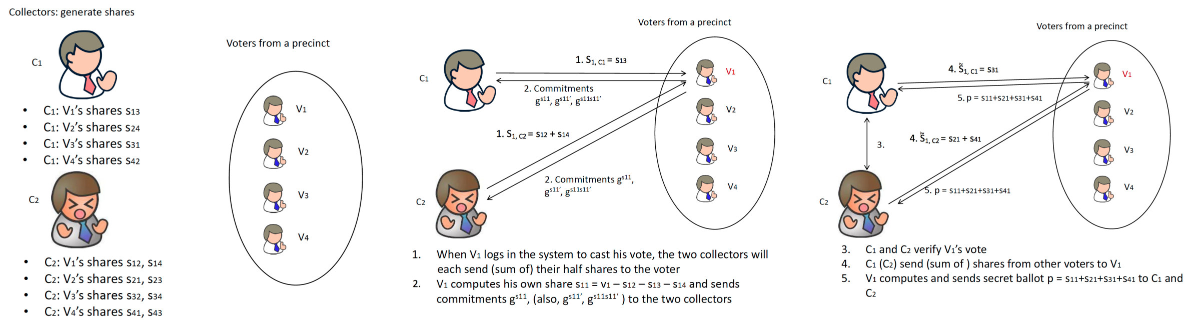
3.3. *Non-Interactive Voting Protocol*

The protocol presented in Section 3.1 is suitable for voting scenarios such as small group election 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 duties 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.

In the interactive voting, a voter depends on other voters' shares in order to cast his ballot. However, it is not practical to require all voters to interact with each other during vote-casting for a large group of voters. Fortunately, our (n, n)-SS based voting protocol can be designed to allow voters to vote non-interactively. Table 2 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 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 2, with $\hat{s_{i,j}}$ $(i \neq j)$ by C_1 and $\check{s_{i,j}}$ $(i \neq j)$ by C_2. s_{ii} is derived by V_i himself. Specifically, 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). Figure 1 (left) illustrates a case of four 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 2, where $j = 1$ or 2) to this voter. Specifically, C_1 sends S_{i,C_1} (sum of shares in one half of the ith row) to the voter, and C_2 sends S_{i,C_2} (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 send the two collectors his commitments (i.e., $g^{s_{ii}}$, $g^{s'_{ii}}$, $g^{s_{ii}s'_{ii}}$). Figure 1 (middle) shows the communication between collectors and voters. Under the assumption that the two collectors have conflicting interests, 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 $\tilde{S}_{i,C_1}$ (sum of shares in one half of the ith column as shown in Table 2) to the voter, and similarly C_2 sends $\tilde{S}_{i,C_2}$ (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 sends the secret ballot of $s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2}$ to the two collectors, as shown in Figure 1 (right) as an example. (Optionally, the voter can send 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**.



- C_1: V_1's shares s_{13}
- C_1: V_2's shares s_{24}
- C_1: V_3's shares s_{31}
- C_1: V_4's shares s_{42}
- C_2: V_1's shares s_{12}, s_{14}
- C_2: V_2's shares s_{21}, s_{23}
- C_2: V_3's shares s_{32}, s_{34}
- C_2: V_4's shares s_{41}, s_{43}

1. When V_1 logs in the system to cast his vote, the two collectors will each send (sum of) their half shares to the voter
2. V_1 computes his own share $s_{11} = v_1 - s_{12} - s_{13} - s_{14}$ and sends commitments g^{s11}, (also, $g^{s11'}$, $g^{s11s11'}$) to the two collectors

3. C_1 and C_2 verify V_1's vote
4. C_1 (C_2) send (sum of) shares from other voters to V_1
5. V_1 computes and sends secret ballot $p = s_{11}+s_{21}+s_{31}+s_{41}$ to C_1 and C_2

Figure 1. Collectors interact with voters to avoid voters' interaction among themselves: **left**—generate shares for voters; **middle**—interact with a casting voter; **right**—interact with the casting voter.

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 generated. As proven by Theorem 1, any k voters, as long as $k \leq N-2$, 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 3.1 still applies.

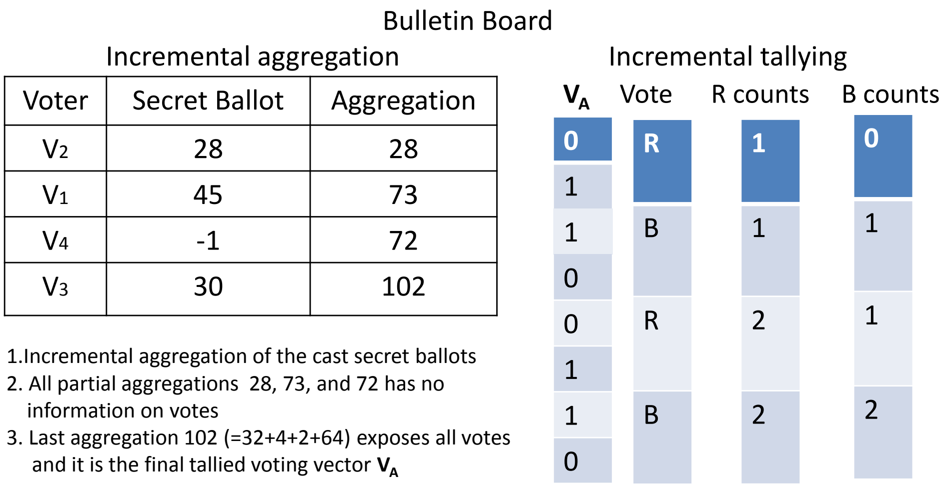
3.4. One Example of Web Based Bulletin Board

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 3 gives an example of 4 voters with their corresponding shares and secret ballot in a case of 2 candidates. Figure 2 illustrates 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.

Table 3. 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)	$17 + 3 + 35 + \underline{9}$	$-1\ (= 7 + (-7) + (-10) + 9)$



Voter	Secret Ballot	Aggregation
V2	28	28
V1	45	73
V4	-1	72
V3	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 $\mathbf{V_A}$

Figure 2. Real Time Bulletin Board.

3.5. Design of a Robust and Efficient LAS

Inspired by the work in [27], we propose a new location anonymization scheme (LAS) that is robust and efficient. Our new scheme solves 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 [27] 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 $\mathbf{L}_i$ (of length $\bar{L}$) with 0s. V_i randomly selects a location $\hat{L}_i$ ($1 \le \hat{L}_i \le \bar{L}$) 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 (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); 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 $l'_i = 10,000$. 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. 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.
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 $max(\mathbf{L}_A) = 1$. 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 3.2) 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 4.5, 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 analysis (in Section 4.5) and simulation result (in Section 5.2).

Notes: (1) Location anonymization, 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 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).

4. Security and Property Analysis

In this section, we demonstrate a few important properties and also analyze the robustness of our protocol.

4.1. Analysis of Main Properties

Here we give main properties of our voting protocol.

Attack-resistance. A random attack against the tallied voting vector with this vector being valid will succeed with the probability of $M^N/2^{MN} = 1/2^{(M-logM)N}$.

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 vector is generated, there must be a certain location containing a vote which does not match what the voter in this location has voted for. Thus, it can be detected and reported by the voter who owns this location.

Completeness (Correctness). All votes are counted correctly in this voting protocol. That is, the aggregated voting vector $\mathbf{V_A}$ of the length N 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, $\mathbf{V}'_\mathbf{A} = \sum_{i=1}^{N} \mathbf{v}'_i$.

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 [30,31], rather than in plain text format/clear vote. In this case, tallying is normally done via homomorphic cryptosystem. Here due to the fundamental principle of homomorphic encryption, voters should be convinced that the final tally is accurate and their votes are accurately 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/anonymized via an anonymous channel or Mix-nets [13,32]. One representative case of this kind is the technique in the paper [33]. 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-nets and blind signature are trustworthy or they can prove their faithful conformation to the protocol via commitment/zero-knowledge proof. 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 [33], our technique allows a voter to verify his vote in plain text format. However, different from [33], the verification in our technique is visibly realized due to transparency and seamless transition from ballots (no information about any vote) to all individual votes (each clear vote is anonymous to any one except the vote's owner). No gap exists and no trustworthy assumptions are required.

Anonymity. The protocol preserves anonymity if no more than $N-2$ voters collude. This claim follows the proof of Theorem 1. 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 revised 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 [25]. 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

$$\begin{aligned} g^{r_1+r'_2+r'_1+r_2} &= g^{S_{i,C_1}S'_{i,C_2}} \times g^{S'_{i,C_1}S_{i,C_2}} \\ &= (g^{S_{i,C_2}})^{S'_{i,C_1}} \times (g^{S'_{i,C_2}})^{S_{i,C_1}} \end{aligned} \tag{1}$$

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 $j = 0, 1, \ldots, L-1$, and constructs $g^{\widehat{S_{i,C_2}}}$ and $g^{\widehat{S'_{i,C_2}}}$ based on Equations (2) and (3) respectively.

$$g^{S_{i,C_2}} = g^{v_i - s_{ii} - S_{i,C_1}} = g^{v_i}(g^{s_{ii}})^{-1}g^{-S_{i,C_1}} \tag{2}$$

$$g^{S'_{i,C_2}} = g^{v'_i - s'_{ii} - S'_{i,C_1}} = g^{v'_i}(g^{s'_{ii}})^{-1}g^{-S'_{i,C_1}} \tag{3}$$

C_1 then verifies if $(g^{\widehat{S_{i,C_2}}})^{S'_{i,C_1}} \times (g^{\widehat{S'_{i,C_2}}})^{S_{i,C_1}}$ (corresponding to the right hand side RHS of Equation (1)) equals to $g^{r_1+r'_2+r'_1+r_2}$ (the left hand side LHS of Equation (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 revised Sub-protocol 1 presented here, P_1 contains a random value of $r_1 + r'_1$, and similarly, P_2 has $r_2 + r'_2$. C_1 can compute either $P_2 \times g^{r_1+r'_1}$ as shown in LHS of Equation (4), or $P_1 \times P_2$ as shown in LHS of Equation (5) to get rid of the randomness. C_1 then guesses v_i as before and plugs it into RHS of Equation (4) or (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 brute-force search. Vote anonymity is preserved.

$$
\begin{aligned}
P_2 \times g^{r_1+r_1'} &= g^{S_{i,C_2}(s_{ii}'+S_{i,C_1}'+S_{i,C_2}')+S_{i,C_2}'(s_{ii}+S_{i,C_1})} = g^{S_{i,C_2}v_i'+S_{i,C_2}'(s_{ii}+S_{i,C_1})} \\
&= g^{(v_i-s_{ii}-S_{i,C_1})v_i'+(v_i'-s_{ii}'-S_{i,C_1}')(s_{ii}+S_{i,C_1})} \\
&= g^{v_i\times v_i'-s_{ii}s_{ii}'-s_{ii}S_{i,C_1}'-s_{ii}'S_{i,C_1}-S_{i,C_1}S_{i,C_1}'}
\end{aligned} \tag{4}
$$

$$
P_1 \times P_2 = g^{(s_{ii}+S_{i,C_1}+S_{i,C_2})\times(s_{ii}'+S_{i,C_1}'+S_{i,C_2}')-s_{ii}s_{ii}'} = g^{v_i\times v_i'-s_{ii}s_{ii}'} \tag{5}
$$

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 [34]. 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.

4.2. 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 $L_c^{i'}$, 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

$$
\begin{aligned}
v_i &= 2^{(L-L_c^i)} + 2^{(L-L_c^{i'})}, v_i' = 2^{(L_c^i-1)} + 2^{(L_c^{i'}-1)}, \\
v_i \times v_i' &= 2^{L-1} + 2^{L-1} + 2^{L-L_c^i+L_c^{i'}-1} + 2^{L-L_c^{i'}+L_c^i-1}.
\end{aligned}
$$

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 ${L_c^i}'$ which is set to 1 and ${L_c^i}' \neq L_c^i$. 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 (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). 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 0 s in the voting vector (arranged in an $N \times M$ 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 $s_{ii} + \tilde{S}_{i,C_1} + \tilde{S}_{i,C_2} \neq p_i$ ($s_{ii}' + \tilde{S}_{i,C_1}' + \tilde{S}_{i,C_2}' \neq p_i'$), 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 conflicts of interest. 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 have discussed earlier in this section. 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 0 s, 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.

4.3. 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.

4.3.1. Robustness against Collusion among Voters

By Theorem 1, the protocol is robust against collusions among voters to infer vote information (passive adversaries) as long as no more than $N-2$ voters 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 4.2.

The analysis above applies to both the interactive protocol and the non-interactive protocol.

4.3.2. Robustness against Collusion among Voters and a Collector

In the interactive protocol, one collector has only a subset of any voter's shares, so the discussion about passive adversaries in Section 4.3.1 still holds here. That is, if no more than $N-2$ voters collude with a collector, no information of votes can be disclosed before the final tally is done.

In the non-interactive protocol, the situation is slightly different. Since collectors generate shares for each voter, they seem to be more powerful than in the interactive protocol. However, all shares of an individual voter are jointly generated by two collectors as shown in Table 2, with each creating only half of shares. The property of (n,n)-SS still applies here. As long as no more than $N-2$ voters collude with a collector, the robustness against collusion among voters and a collector to infer vote information still holds.

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 discussed earlier in this section. 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.

4.4. 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 4.3.

4.5. Robustness of Location Anonymization

The analysis in Section 4.2 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 $g^{g^{10^{\tilde{L}-\hat{L}_i}}}/g^{s_{ii}} = Q$ ($g^{g^{10^{\hat{L}_i-1}}}/g^{s'_{ii}} = Q'$). 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 2.1 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.

5. Complexity Analysis and Simulation

We provide complexity analysis and then simulation results.

5.1. 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. The 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 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 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 $O(N^2)$ shares from voters in the interactive protocol with the cost of $O(N^2T)$, or sending sums of shares 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 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 $O(N^2T) + (O(\tilde{T}) + O(T))N$ 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 the computation costs of Sub-protocol 1 and Sub-protocol 2.

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 P_j involves tailed STPM; and (3) computing $g^{s_{ii}s'_{ii}} \times P_1 \times P_2$ costs $O(T^2)$. Computing

P_j consists of obtaining r_j (r'_j) with 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.

LAS uses similar mechanisms of the voting protocol during each round. Thus for each round, we obtain similar complexity. Roughly, the message length T in LAS is $O(\bar{L})$.

5.2. Simulation Result

The results presented here are from our protocol simulation implemented in Java. The experiments were carried out on a computer with a 1.87 GHz CPU and 32 GB of memory. 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.

Figure 3 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 Figure 3, the number of rounds needed for completing location anonymization is relatively stable for different N under a given ratio $\bar{L}/N$.

Figure 4 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 Figure 3. 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 sufficient for completing location anonymization. For example, with 1000 voters, it took no more than 0.05 s to anonymize voters' locations. This demonstrates the efficiency of the proposed LAS.

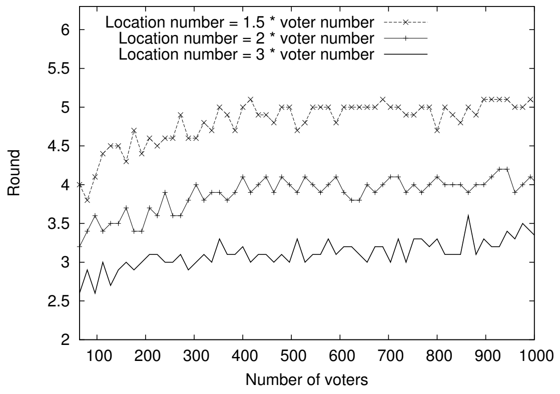


Figure 3. Location Anonymity Scheme (LAS): number of rounds needed for location anonymization.

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.

Figures 5 and 6 show the computation time of Sub-protocol 1 and Sub-protocol 2, respectively. Sub-protocol 1 was dominated by tailored 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 much greater computing power.

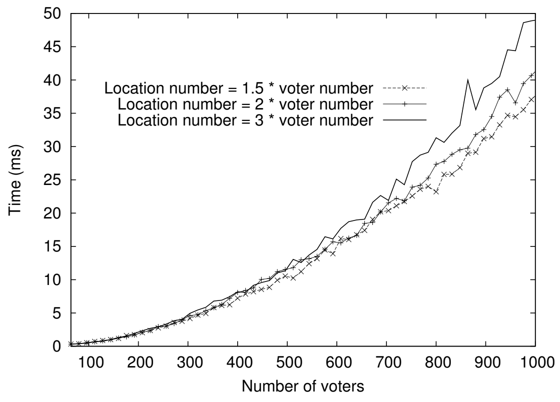


Figure 4. LAS: time needed for location anonymization per voter.

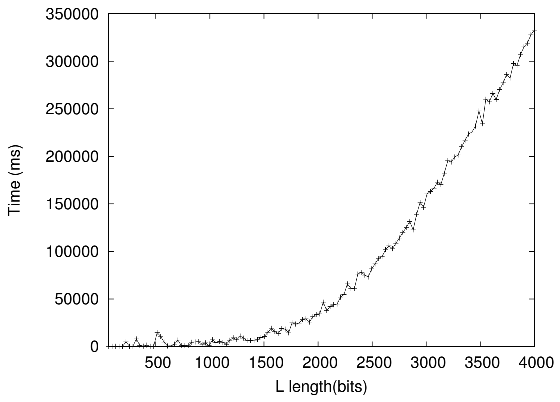


Figure 5. Collectors run Sub-protocol 1 in TP3 against one voter.

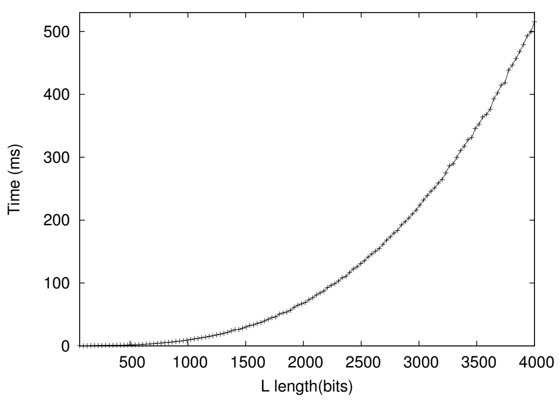


Figure 6. Collectors run Sub-protocol 2 in TP3 against one voter.

Figure 7 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 $NL = O(L^2)$.

The simulation results confirm the performance analysis in Section 5.1. Most operations are quite efficient. For example, when $L = 4000$ (and $N = 2000$), collecting and tallying votes took only 0.005 s. For the in-process enforcement protocol however, it took the collectors 332 s to complete Sub-protocol 1 and 0.515 s 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.

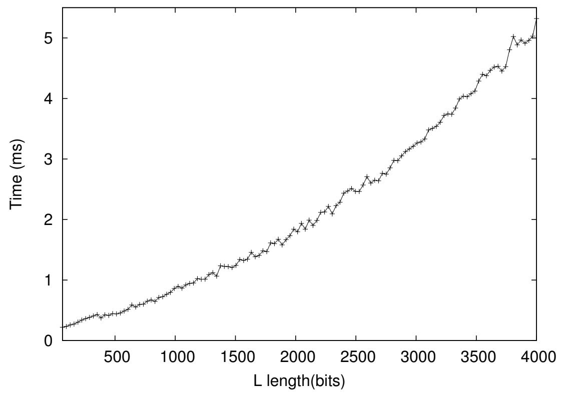


Figure 7. One collector collects/tallies votes.

6. 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 [7–12,17,30,35–43].

Cryptographic technique has been an indispensable component of most online voting systems. A variety of cryptographic techniques, such as mix-nets, blind signature, homomorphic encryption, zero-knowledge proof, and secret sharing, are deployed in electronic voting protocols to secure voter's vote. The first e-voting scheme proposed by Chaum [44] in 1981 utilizes anonymous channels (i.e., mix-nets). Additional schemes [7,13,45–50] based on mix-nets are proposed afterwards with various optimization. For example, Aditya et al. [49] improve the efficiency of Lee et al.'s scheme [46] through modified optimistic mix-nets. The scheme in [7] uses two kinds of mix-nets to prevent vote updating from being detected by coercers. However, due to the usage of mix-nets, transparency cannot be guaranteed.

A blind signature allows an authority to sign an encrypted message without knowing the message's context [14,33,50–55]. 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. [56] 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-nets is proposed in [52]. Similarly, blind signature is used in a debate voting [55] with messages of varying length where anonymous channel is assumed.

Voting schemes based on homomorphic encryption can trace back to the seminal works by Benaloh [16,57] and later development in efficiency [31,58], and receipt-freeness [15,59–61]. 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. [62] utilizes additively homomorphic public keys on bilinear groups with 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 [15,16,57,59–61]. Some schemes [31,58] utilize Shamir's threshold secret sharing [63], while some [64] are based on Chinese remainder theorem. In contrast, ours is based on a simplified (n, n)-SS scheme. In existing voting schemes, the 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 [15,16,57–60,65]; and (b) to pool their shares together to recover the encrypted or masked tally result [31,64]. Instead, in our scheme, the secret sharing is used among voters to share their secret votes and then recover their open yet anonymous votes.

Particularly, some existing protocols require physical settings such as voting booths [23], a tamper resistant randomizer [46,60,66,67], or specialized smart cards [68]. 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 [45,69] 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 [70,71] 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 [72–75] allows the voter to obtain a confirmation code from the paper ballot. Through the confirmation code, the voter can verify 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 [23] is a (physical) split ballot voting mechanism by splitting the trust between two conflict-of-interest 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 [76] 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 [77,78] 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 [79,80] 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 has not considered how individual or universal verification is done.

Scytl [81–84] 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 devices.

In the ADDER [85] 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 [85] 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.

Unfortunately, due to the strict and conflicting e-voting requirements [5], there is not any scheme/system currently satisfying all voting properties at the same time [42]. Security weakness is found even in highly referenced voting schemes [86]. The papers [87,88] 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 [87] and a general definition of verifiability in [88].

Comparison with Helios. Helios [89] implements Benaloh's vote-casting approach [90] on the Sako-Kilian mix-nets [91]. 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, integrity, and open source.

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 new 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 new protocol is visibly transparent. *Third*, in terms of voter assurance, the transition from ballots to plain votes in Helios involves mix-net (shuffling and re-encryption) and decryption. In contrast, such transition in our new protocol is seamless and viewable. In addition, the voter can conduct self-tallying. Thus, voter assurance in our new 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 new protocol is straightly based on simple addition and viewable verification. Thus, accuracy of the tallying result in our new protocol is self-evident 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 involves exponentiations (decryption). However, our ballot generation and tallying need only modular subtractions and additions. Thus, our new protocol is more efficient.

Besides Zeus [20] and Helios 2.0 [92], there are some variants of Helios such as BeleniosRF [93]. BeleniosRF is built upon Belenios. It introduces signatures on randomizable ciphertexts to achieve receipt-freeness. A voting authority is assumed to be trustworthy.

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 [94]. One interesting application domain of distributed e-voting is boardroom voting [21]. 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 [22,95] and our interactive voting protocol is similar to them too. In all these protocols including ours, tallying is called self-tallying: each voter can incrementally aggregate ballots/votes themselves by either summing the votes [95]) (as well as ours) or multiplying the ballots [22]) (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 [21], in terms of vote casting, the protocol in [22] needs $4N + 10$ exponentiations per voter and the protocol in [95] needs $2N + 2$. However, our interactive voting protocol needs only 6. In terms of tallying [21], the protocol in [22] needs $11N - 11 + \binom{N+M}{M}/2$ and the protocol in [95] 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 beyond 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 [22], plain votes are not viewable, and in [95], even though plain votes are viewable but the voter does not know which one is his because the shuffling process changes the correspondence between the initial ballots and (decrypted) individual plain votes.

7. Discussion of Scalability and Hierarchical Design

In this section, we discuss about 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 5.2. The results showed 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 [96]. 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.

8. Conclusions and Future Work

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 particular, the interactive protocol is suitable for election within a small group 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.

As the future work, we plan to further improve our protocol, focusing particularly on receipt freeness and voter coercion. They are the most challenging problems in e-voting, compared to other requirements. As pointed out in [20], e-voting is inherently coercible. Specifically in our protocol, since the tallied voting vector $\mathbf{V_A}$ contains all individual votes, a voter's location can potentially become a self-claimed receipt and the voter can exploit it for vote selling.

To prevent vote selling, we plan to have collectors jointly shuffle/shift the locations together with votes in $\mathbf{V_A}$ randomly and then publish the skewed $\mathbf{V_A}$. In this case, individual verification cannot be done visually but a voter can perform 1 out of N oblivious transfer with collectors for verification. To prevent voter coercion, we plan to let each voter have multiple locations, for example, one real location and one fake location. The coerced voter can cast the vote as what a coercer asked for using the fake location during the presence of the coercer and then cast his real vote in the real location afterward. To prevent both, some combination of these can be deployed, or solutions based on secure two-party computation can be developed.

Recent research by Grewal et al. [6] acknowledges the toughness of the voter coercion issue, and they propose 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 [97].

Author Contributions: X.Z. initiated the design of the protocol and led the writing and proof-reading of the paper. H. L., F. L., W.P., and Y. S. joined the discussion, security proof, enhancement, and experiment of the protocol. H.L. contributed significantly to the writing and editing of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wikipedia. Help America Vote Act, The Free Encyclopedia, 2013. Available online: https://en.wikipedia.org/wiki/Help_America_Vote_Act (accessed on 1 July 2013).
2. Wikipedia. The U.S. Election Assistance Commission, 2013. Available online: https://www.eac.gov/about/help-america-vote-act/ (accessed on 1 July 2013).
3. Commission, T.U.E.A. 2008 Election Administration and Voting Survey. Available online: http://www.eac.gov/ (accessed on 1 September 2009).
4. Campbell, B. The Usability Implications of Long Ballot Content for Paper, Electronic, and Mobile Voting Systems. Ph.D. Thesis, RICE University, Houston, TX, USA, 2014.
5. Chevallier-Mames, B.; Fouque, P.A.; Pointcheval, D.; Stern, J.; Traoré, J. On some incompatible properties of voting schemes. In *Towards Trustworthy Elections*; Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.A., Benaloh, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 191–199.
6. Grewal, G.S.; Ryan, M.D.; Bursuc, S.; Ryan, P.Y. Caveat Coercitor: Coercion-evidence in electronic voting. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 367–381.
7. Araújo, R.; Barki, A.; Brunet, S.; Traoré, J. Remote Electronic Voting Can Be Efficient, Verifiable and Coercion-Resistant. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, 26 February 2016, Revised Selected Papers*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 224–232.
8. Li, C.; Hwang, M.; Lai, Y. A Verifiable Electronic Voting Scheme over the Internet. In Proceedings of the Sixth International Conference on Information Technology: New Generations (ITNG '09), Las Vegas, NV, USA, 27–29 April 2009; pp. 449–454.
9. Howlader, J.; Nair, V.; Basu, S.; Mal, A. Uncoercibility In E-Voting and E-Auctioning Mechanisms Using Deniable Encryptiony. *Int. J. Netw. Secur. Appl.* **2011**, *3*, 97–109.
10. Lee, C.; Chen, T.; Lin, S.; Hwang, M. A New Proxy Electronic Voting Scheme Based on Proxy Signatures. In *Lecture Notes in Electrical Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 164, pp. 3–12.
11. Spycher, O.; Koenig, R.; Haenni, R.; Schläpfer, M. A New Approach towards Coercion-Resistant Remote E-Voting in Linear Time. In Proceedings of the International Conference on Financial Cryptography and Data Security, Kralendijk, Bonaire, 27 February–2 March 2012; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7035, pp. 182–189.
12. Clarkson, M.; Chong, S.; Myers, A. Civitas: Toward a secure voting system. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 18–22 May 2008; pp. 354–368.
13. Juels, A.; Catalano, D.; Jakobsson, M. Coercion-resistant electronic elections. In Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, Alexandria, VA, USA, 7 November 2005; ACM: New York, NY, USA, 2005; pp. 61–70.

14. Chaum, D. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT '88), Davos, Switzerland, 25–27 May 1988; pp. 177–182.
15. Rjaskova, Z. Electronic Voting Schemes. Ph.D. Thesis, Comenius University, Bratislava, Slovakia, 2002.
16. Benaloh, J. Verifiable Secret Ballot Elections. Ph.D. Thesis, Yale University, New Haven, CT, USA, 1987.
17. Jones, D.; Simons, B. *Broken Ballots: Will Your Vote Count?* The University of Chicago Press: Chicago, IL, USA, 2012.
18. Shamos, M.I. Electronic Voting Records—An Assessment. In *Developing an Analysis of Threats to Voting Systems: Preliminary Workshop Summary*; NIST: Gaithersburg, MD, USA, 2004; pp. 227–251.
19. McGaley, M.; McCarthy, J. Transparency and e-voting: Democratic vs. commercial interests. *Proc. GI* **2012**, *47*, 153–163.
20. Tsoukalas, G.; Papadimitriou, K.; Louridas, P.; Tsanakas, P. From Helios to Zeus. In Proceedings of the 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '13), Washington, DC, USA, 12–13 August 2013; pp. 1–17.
21. Kulyk, O.; Neumann, S.; Budurushi, J.; Volkamer, M.; Haenni, R.; Koenig, R.; Bergen, P. Efficiency Evaluation of Cryptographic Protocols for Boardroom Voting. In Proceedings of the 2015 10th International Conference on Availability, Reliability and Security (ARES), Toulouse, France, 24–27 August 2015; pp. 224–229.
22. Khader, D.; Smyth, B.; Ryan, P.Y.; Hao, F. A fair and robust voting system by broadcast. In Proceedings of the EVOTE'12: 5th International Conference on Electronic Voting, Bregenz, Austria, 11–14 July 2012; pp. 1–13.
23. Moran, T.; Naor, M. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.* **2010**, *13*, 16:1–16:43.
24. Wikipedia. Vote Counting System 2012. Available online: https://en.wikipedia.org/wiki/Vote_counting (accessed on 21 June 2012).
25. Zou, X.; Li, H.; Sui, Y.; Peng, W.; Li, F. Assurable, Transparent, and Mutual Restraining E-Voting Involving Multiple Conflicting Parties. In Proceedings of the IEEE INFOCOM, Toronto, ON, Canada, 27 April–2 May 2014; pp. 136–144.
26. Stinson, D.R. *Cryptography: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 1995; Chapter 11, pp. 330–331.
27. Zhao, X.; Li, L.; Xue, G.; Silva, G. Efficient anonymous message submission. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA , 25–30 March 2012; pp. 2228–2236.
28. Samet, S.; Miri, A. Privacy preserving ID3 using Gini Index over horizontally partitioned data. In Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '08), Doha, Qatar, 31 March–4 April 2008; pp. 645–651.
29. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654.
30. Benaloh, J.; Tuinstra, D. Receipt-free secret-ballot elections. In Proceedings of the 26th Annual ACM Symposium on Theory of Computing, Montréal, QC, Canada, 23–25 May 1994; ACM: New York, NY, USA, 1994; pp. 544–553.
31. Schoenmakers, B. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '99), Santa Barbara, CA, USA, 15–19 August 1999; pp. 148–164.
32. Clark, J.; Hengartner, U. Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance. In *Financial Cryptography and Data Security*; Lecture Notes in Computer Science; Danezis, G., Ed.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7035, pp. 47–61.
33. Fujioka, A.; Okamoto, T.; Ohta, K. A practical secret voting scheme for large scale elections. In *Advances in Cryptology—AUSCRYPT '92*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 244–251.
34. Spafford, E.H. Voter Assurance. In *Voting Technologies*; National Academy of Engineering: Washington, DC, USA, 2007; pp. 28 –34.
35. Clarkson, M.; Chong, S.; Myers, A. *Civitas: Toward a Secure Voting System*; Technical Report; Cornell University: Ithaca, NY, USA, 2008.
36. Shirazi, F.; Neumann, S.; Ciolacu, I.; Volkamer, M. Robust electronic voting: Introducing robustness in Civitas. In Proceedings of the International Workshop on REVOTE, Trento, Italy, 29 August 2011; pp. 47–55.
37. Volkamer, M.; Grimm, R. Determine the Resilience of Evaluated Internet Voting Systems. In Proceedings of the International Workshop on REVOTE, Atlanta, GA, USA, 31 August 2009; pp. 47–54.

38. Li, C.; Hwang, M. Security Enhancement of Chang-Lee Anonymous E-Voting Scheme. *Int. J. Smart Home* **2012**, *6*, 45–52.
39. Staff, C. Seven principles for secure e-voting. *Commun. ACM* **2009**, *52*, 8–9.
40. Epstein, J. Electronic voting. *IEEE Comput.* **2007**, *40*, 92–95.
41. Kusters, R.; Truderung, T.; Vogt, A. Clash Attacks on the Verifiability of E-Voting Systems. In Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–23 May 2012; pp. 395–409.
42. Fouard, L.; Duclos, M.; Lafourcade, P. Survey on Electronic Voting Schemes. Available online: http://www-verimag.imag.fr/~duclos/paper/e-vote.pdf (accessed on 1 July 2013).
43. Benaloh, J.; Byrne, M.; Kortum, P.; McBurnett, N.; Pereira, O.; Stark, P.; Wallach, D. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. *arXiv* **2012**, arXiv:1211.1904.
44. Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **1981**, *24*, 84–90.
45. Chaum, D.; Ryan, P.; Schneider, S. A practical voter-verifiable election scheme. In Proceedings of the 10th European Conference on Research in Computer Security (ESORICS '05), Milan, Italy, 12–14 September 2005; pp. 118–139.
46. Lee, B.; Boyd, C.; Dawson, E.; Kim, K.; Yang, J.; Yoo, S. Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Korea, 27–28 November 2003; Volume 2971, pp. 245–258.
47. Weber, S. A Coercion-Resistant Cryptographic Voting Protocol—Evaluation and Prototype Implementation. Master's Thesis, Darmstadt University of Technology, Darmstadt, Germany, 2006.
48. Chaum, D. Secret-ballot receipts: True voter-verifiable elections. *IEEE Secur. Priv.* **2004**, *2*, 38 – 47.
49. Aditya., R.; Lee, B.; Boyd, C.; Dawson, E. An Efficient Mixnet-Based Voting Scheme Providing Receipt-Freeness. In Proceedings of the International Conference on Trust and Privacy in Digital Business, Zaragoza, Spain, 30 August–1 September 2004; Volume 3184, pp. 152–161.
50. Okamoto, T. Receipt-free electronic voting schemes for large scale elections. In Proceedings of the International Workshop on Security Protocols, Paris, France, 7–9 April 1997; Springer: Berlin/Heidelberg, Germany, 1998; pp. 25–35.
51. Radwin, M. An Untraceable, Universally Verifiable Voting Scheme. Available online: https://www.researchgate.net/publication/2867316_An_Untraceable_Universally_Verifiable_Voting_Scheme (accessed on 16 August 2017).
52. Ohkubo, M.; Miura, F.; Abe, M.; Fujioka, A.; Okamoto, T. An Improvement on a Practical Secret Voting Scheme. In Proceedings of the 2nd International Workshop on Information Security (ISW '99), Kuala Lumpur, Malaysia, 6–7 November 1999; pp. 225–234.
53. Kim, K.; Kim, J.; Lee, B.; Ahn, G. Experimental Design of Worldwide Internet Voting System Using PKI. In Proceedings of the SSGRR, LÁquila, Italy, 6–12 August 2001; pp. 1–7.
54. Boyd, C. A new multiple key cipher and an improved voting scheme. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '89), Houthalen, Belgium, 10–13 April 1989; pp. 617–625.
55. García, D.L. A flexible e-voting scheme for debate tools. *Comput. Secur.* **2016**, *56*, 50–62.
56. Chow, S.; Liu, J.; Wong, D. Robust Receipt-Free Election System with Ballot Secrecy and Verifiability. In Proceedings of the 16th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA, USA, 8–11 February 2008; pp. 81–94.
57. Benaloh, J.; Yung, M. 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 (PODC '86), Calgary, AB, Canada, 11–13 August 1986; pp. 52–62.
58. Cramer, R.; Gennaro, R.; Schoenmakers, B. A secure and optimally efficient multi-authority election scheme. In Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT '97), Konstanz, Germany, 11–15 May 1997; pp. 103–118.
59. Hirt, M.; Sako, K. Efficient receipt-free voting based on homomorphic encryption. In Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000; pp. 539–556.

60. Lee, B.; Kim, K. Receipt-free Electronic Voting through Collaboration of Voter and Honest Verifier. In Proceedings of the JW-ISC, Naha, Japan, 25–26 January 2000; pp. 101–108.
61. Adewole, A.P.; Sodiya, A.S.; Arowolo, O.A. A Receipt-Free Multi-Authority E-Voting System. *Int. J. Comput. Appl.* **2011**, *30*, 15–23.
62. Kiayias, A.; Zacharias, T.; Zhang, B. DEMOS-2: Scalable E2E Verifiable Elections without Random Oracles. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; ACM: New York, NY, USA, 2015; pp. 352–363.
63. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613.
64. Iftene, S. General Secret Sharing Based on the Chinese Remainder Theorem with Applications in E-Voting. *Electron. Notes Theor. Comput. Sci.* **2007**, *186*, 67–84.
65. Cramer, R.; Franklin, M.; Schoenmakers, B.; Yung, M. Multi-Authority Secret-Ballot Elections with Linear Work. In *Advances in Cryptology— EUROCRYPT '96*; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1070, pp. 72–83.
66. Baudron, O.; Fouque, P.A.; Pointcheval, D.; Stern, J.; Poupard, G. Practical multi-candidate election system. In Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC '01), Newport, RI, USA, 26–28 August 2001; pp. 274–283.
67. Hirt, M. Receipt-free K-out-of-L voting based on elgamal encryption. In *Towards Trustworthy Elections*; Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.A., Benaloh, J., Eds.; Springer Berlin/Heidelberg, Germany, 2010; pp. 64–82.
68. Han, W.; Chen, K.; Zheng, D. Receipt-Freeness for Groth's E-Voting Schemes. *J. Inf. Sci. Eng.* **2009**, *25*, 517–530.
69. Ryan, P.; Peacock, T. *Prêt à Voter: A Systems Perspective*; Technical Report CS-TR-929; University of Newcastle: Newcastle, Australia, 2005.
70. Rivest, R. The ThreeBallot Voting System. Avaialable online: http://theory.lcs.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf (accessed on 1 July 2013).
71. Rivest, R.; Smith, W. Three voting protocols: ThreeBallot, VAV, and Twin. In Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology (EVT '07), Boston, MA, USA, 6–10 August 2007; Volume 16, pp. 1–14.
72. Fisher, K.; Carback, R.; Sherman, A. Punchscan: Introduction and system definition of a high-integrity election system. In Proceedings of Workshop on Trustworthy Elections, Cambridge, UK, 29–30 June 2006; pp. 19–29.
73. Popoveniuc, S.; Hosp, B. An introduction to Punchscan. In Proceedings of the IAVoSS Workshop on Trustworthy Elections (WOTE 2006), Cambridge, UK, 29–30 June 2006; pp. 28–30.
74. Chaum, D.; Essex, A.; Carback, R.; Clark, J.; Popoveniuc, S.; Sherman, A.; Vora, P. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Secur. Priv.* **2008**, *6*, 40–46.
75. Chaum, D.; Carback, R.; Clark, J.; Essex, A.; Popoveniuc, S.; Rivest, R.; Ryan, P.; Shen, E.; Sherman, A. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. *EVT* **2008**, *8*, 1–13.
76. Bohli, J.; Müller-Quade, J.; Röhrich, S. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In Proceedings of the International Conference on E-Voting and Identity, Bochum, Germany, 4–5 October 2007; pp. 111–124.
77. Sandler, D.; Wallach, D. Casting votes in the auditorium. In Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology (EVT '07), Boston, MA, USA, 6–10 August 2007; pp. 1–15.
78. Sandler, D.; Derr, K.; Wallach, D. VoteBox: A tamper-evident, verifiable electronic voting system. In Proceedings of the USENIX Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; Volume 4, pp. 349–364.
79. Cross, E.V., II.; McMillian, Y.; Gupta, P.; Williams, P.; Nobles, K.; Gilbert, J.E. Prime III: A user centered voting system. In Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI '07), San Jose, CA, USA, 28 April–3 May 2007; pp. 2351–2356.
80. Dawkins, S.; Sullivan, T.; Rogers, G.; Cross, E.V., II; Hamilton, L.; Gilbert, J.E. Prime III: An innovative electronic voting interface. In Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI '09), Sanibel Island, FL, USA, 8–11 February 2009; pp. 485–486.

81. Scytl. White paper: Auditability and Voter-Verifiability for Electronic Voting Terminals. Available online: http://www.scytl.com/images/upload/home/PNYX.VM_White_Paper.pdf (accessed on 1 July 2013).
82. SCYTL. *Pnyx.VM: Auditability and Voter-Verifiability for Electronic Voting Terminals*; SCYTL: Barcelona, Spain, 2004.
83. SCYTL. *Pnyx.core: The Key to Enabling Reliable Electronic Elections*; SCYTL: Barcelona, Spain, 2005.
84. Clarkson, M.; Hay, B.; Inge, M.; Wagner, D.; Yasinsac, A. Software Review and Security Analysis of Scytl Remote Voting Software. Available online: http://www.cs.cornell.edu/~clarkson/papers/scytl-odbp.pdf. (accessed on 16 August 2017).
85. Kiayias, A.; Korman, M.; Walluck, D. An Internet voting system supporting user privacy. In Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06), Miami Beach, FL, USA, 11–15 December 2006; IEEE Computer Society: Washington, DC, USA, 2006; pp. 165–174.
86. Karlof, C.; Sastry, N.; Wagner, D. Cryptographic Voting Protocols: A Systems Perspective. In Proceedings of the 14th conference on USENIX Security Symposium, Baltimore, MD, USA, 31 July–5 August 2005; Volume 5, pp. 33–50.
87. Bernhard, D.; Cortier, V.; Galindo, D.; Pereira, O.; Warinschi, B. SoK: A comprehensive analysis of game-based ballot privacy definitions. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–20 May 2015; IEEE Computer Society: Los Alamitos, CA, USA, 2015, pp. 499–516.
88. Cortier, V.; Galindo, D.; Küsters, R.; Müller, J.; Truderung, T. SoK: Verifiability Notions for E-Voting Protocols. In Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P '16), San Jose, CA, USA, 23–25 May 2016; pp. 779–798.
89. Adida, B. Helios: Web-based open-audit voting. In Proceedings of the USENIX Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; USENIX Association: Berkeley, CA, USA, 2008; Volume 17, pp. 335–348.
90. Benaloh, J. Simple verifiable elections. In Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (EVT/WOTE. USENIX), Vancouver, BC, Canada, 1 August 2006; pp. 1–10.
91. Sako, K.; Kilian, J. Receipt-free mix-type voting scheme. In *Advances in Cryptology—UROCRYPT '95*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 393–403.
92. Adida, B.; De Marneffe, O.; Pereira, O.; Quisquater, J. 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), Montreal, QC, Canada, 10–11 August 2009.
93. Cortier, V.; Fuchsbauer, G.; Galindo, D. BeleniosRF: A Strongly Receipt-Free Electronic Voting Scheme. Cryptology ePrint Archive, Report 2015/629, 2015. Available online: http://eprint.iacr.org/2015/629 (accessed on 16 August 2017).
94. Nakajima, A. Decentralized voting protocols. In Proceedings of the International Symposium on Autonomous Decentralized Systems, Kawasaki, Japan, 30 March–1 April 1993; IEEE Computer Society Press: Los Alamitos, CA, USA, 1993; pp. 247–254.
95. Kulyk, O.; Neumann, S.; Feier, C.; Volkamer, M.; Koster, T. Electronic voting with fully distributed trust and maximized flexibility regarding ballot design. In Proceedings of the 6th International Conference on Electronic Voting (EVOTE), Lochau, Austria, 29–31 October 2014; IEEE: Piscataway, NJ, USA, pp. 1–10.
96. Commission, T.U.E.A. Polling Places 2004 General Election: EAC Election Day Survey. Available online: https:/electioncenter.org/documents/EAC-2004%20election%20surveyFull_Report_wTables.pdf (accessed on 16 August 2017).
97. Benaloh, J. Rethinking Voter Coercion: The Realities Imposed by Technology. In Proceedings of the 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '13), Washington, DC, USA, 12–13 August 2013; USENIX: Berkeley, CA, USA, 2013; pp. 82–105.