Secure Order Based Voting Using Distributed Tallying

Tamir Tassa[a], Lihi Dery[b], Arthur Zamarin[a],

[a] *The Open University, 1 University Road, Ra'anana, 4353701, Israel*
[b] *Ariel University, 3 Kiryat HaMada Street, Ariel, 40700, Israel*

Abstract

Electronic voting systems have significant advantages in comparison with physical voting systems. One of the main challenges in e-voting systems is to secure the voting process: namely, to certify that the computed results are consistent with the cast ballots and that the voters' privacy is preserved. We propose herein a secure voting protocol for elections that are governed by order-based voting rules. Our protocol, in which the tallying task is distributed among several independent talliers, offers perfect ballot secrecy in the sense that it issues only the required output while no other information on the cast ballots is revealed. Such perfect secrecy, achieved by employing secure multiparty computation tools, may increase the voters' confidence and, consequently, encourage them to vote according to their true preferences. We implemented a demo of a voting system that is based on our protocol and we describe herein the system's components and its operation. Our implementation demonstrates that our secure order-based voting protocol can be readily implemented in real-life large-scale electronic elections.

Keywords: Secure voting, Order-based voting rules, Secret sharing, Multiparty computation

1. Introduction

Electronic voting has significant advantages in comparison with the more commonplace physical voting: it is faster, it reduces costs, it is more sustainable, and in addition, it may increase voters' participation. In recent years, due to the COVID-19 pandemic and the need for social distancing, e-voting platforms have become even more essential. One of the main challenges in holding e-voting is to secure the voting process: namely, to make sure that it

is secure against attempted manipulations so that the computed results are consistent with the cast ballots and that the privacy of the voters is preserved.

The usual meaning of voter privacy is that the voters remain anonymous. Namely, the linkage between ballots and the voters that cast them remains hidden, thus preserving anonymity. However, the final election results and the full tally (i.e., all ballots) are revealed. While such information exposure may sometimes be considered benign or even desired, in some cases it may be problematic. For example, in small-scale elections—such as those for a university senate—publishing the full tally may compromise ballot secrecy. Suppose Bob is a candidate and only a small number of voters participate. If the final tally reveals that Bob received zero votes, and it is known or assumed that Alice supported him, observers (including Bob himself) may infer that Alice did not follow through. Such inference risks may deter voters from expressing their true preferences, either out of fear of social consequences or due to pressure from candidates or peers, particularly in close-knit communities.

In contrast to weaker privacy notions, such as anonymity or partial tally hiding, perfect ballot secrecy ensures that nothing is disclosed except for the final results. The final results could be just the identity of the winner (say, when selecting an editor-in-chief or a prime minister), $K > 1$ winners (e.g., when needing to select K new board members), $K > 1$ winners with their ranking (say, when needing to award first, second and third prizes), or a score for each candidate (as is the case in elections for parliament, where the candidates are parties and the score for each party is the number of seats in the parliament).

In this paper, we present the design of a tally-hiding voting system that ensures perfect ballot secrecy—a concept sometimes referred to as full privacy (see, e.g., [12]). Such secrecy ensures that given any coalition of voters, the protocol does not reveal any information on the ballots beyond what can be inferred from the published results. Such perfect ballot secrecy may reduce the possibility of voters' coercion and increase their confidence that their vote remains secret. Hence, it may encourage voters to vote according to their true preferences.

There are various families of voting rules that can be used in elections. For example, in *score-based* voting rules, a score for each candidate is computed subject to the specifications of the underlying voting rule, and then the winning candidate is the one whose aggregated score is the highest. In this study, we focus on *order-based* (a.k.a *pairwise-comparison*) voting rules, where

the relative order of the candidates is considered by the underlying voting rule [8].

Our contributions. We consider a scenario in which there is a set of voters that hold an election over a given set of candidates, where the election is governed by an order-based voting rule. We devise a fully private protocol for computing the election results. The election output is a ranking of the candidates from which the winning candidate(s) can be determined. Our protocol is lightweight and can be readily implemented in virtual elections.

In order to provide privacy for the voters, it is necessary to protect their private data (i.e. their ballots) from the tallier while still allowing the tallier to perform the needed computations on the ballots in order to output the required election results. The cryptographic tool that is commonly used towards this end is *homomorphic encryption*, namely, a form of encryption that preserves the algebraic structure and thus enables the performance of meaningful computations on the ciphertexts. However, homomorphic encryption has a significant computational overhead. The main idea that underlies our suggested system is to use *distributed* tallying. Namely, our system involves $D > 1$ independent talliers to whom the voters send information relating to their private ballots. With such distributed tallying, it is possible to replace the costly cryptographic protection shield of homomorphic encryption with the much lighter-weight cryptographic shield of secret sharing (all relevant cryptographic background is provided in Section 2). The talliers then engage in specially designed protocols of multiparty computation that allow them to validate that each cast ballot is a legitimate ballot (in the sense that it complies with the specifications of the underlying voting rule) and then to compute from the cast ballots the required election results, while still remaining totally oblivious to the content of those ballots. A high-level description of our system is presented in Figure 1. The first phase is the voting phase: the voters send to each of the talliers messages (shares) relating to their secret ballots. In the second phase, the talliers communicate among themselves in order to validate each of the incoming ballots (which remain secret to them) and then, at the end of the day, perform the tallying over all legal ballots according to the underlying rule. Finally, the talliers broadcast the results back to the voters.

The bulk of this study is devoted to two order-based voting rules – Copeland [15], and Maximin [47] (a.k.a Kramer-Simpson), which are formally defined in Section 3.1. In Appendix Appendix B we describe two

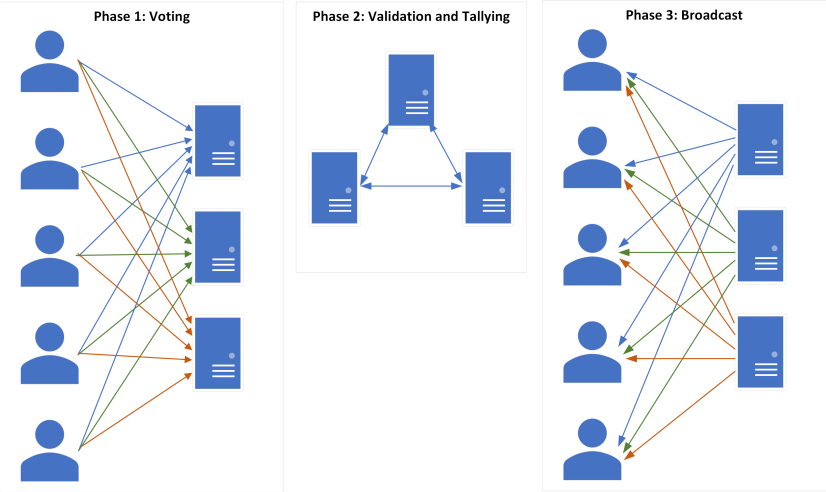

Figure 1: A high-level description of the protocol.

other rules in this family, Kemeny-Young [31, 51] and Modal Ranking [10], and describe the extension of our methods for those rules as well.

The paper is structured as follows. In Section 2 we explain the cryptographic principles of distributed tallying. In Section 3 we present our secure voting protocols for Copeland and Maximin rules. We focus there on the case where voters must determine a strict ranking of the candidates. Then, in Section 4, we extend our discussion to the more involved case in which the voters' rankings may include ties between some of the candidates. In Section 5 we describe an implementation of a secure voting system that is based on our protocols; our implementation is in Python and is open source. In Section 6 we survey related work. Lastly, Section 7 provides concluding comments, as well as a description of an implementation of our presented protocol and system in the Gentoo[1] council elections.

The Appendix includes additional discussions. In Appendix A we describe cryptographic sub-protocols that we use in our protocol, while in Appendix B we discuss secure protocols for two other order-based voting rules — Kemeny-Young and Modal Ranking.

[1]https://www.gentoo.org/

2. Distributed tallying

A key principle of our protocol is distributing the tallying task among multiple independent talliers. This distribution is supported by secret sharing [44] and secure multiparty computation (MPC) [50]. In Section 2.1 we give a brief recap of secret sharing. Then, in Section 2.2 we describe the distributed tallying on which our protocol is based.

Our protocol utilizes MPC sub-protocols that perform secure computations over secret shares. We treat those sub-protocols as black boxes. We describe their inputs and outputs in Section 2.2. Readers seeking a deeper understanding of these MPC components may refer to Appendix A.

2.1. Secret sharing

Secret sharing schemes [44] enable distributing a secret s among a set of parties, $\mathbf{T} = \{T_1, \ldots, T_D\}$. Each party, T_d, $d \in [D] := \{1, \ldots, D\}$, is given a random value s_d, called *a share*, that relates to the secret s. Those shares satisfy the following two conditions: (a) s can be reconstructed only by combining the shares given to specific *authorized* subsets of parties, and (b) shares belonging to unauthorized subsets of parties reveal zero information on s. In *threshold* secret sharing there is some threshold $D' \leq D$ and then the authorized subsets are those of size at least D'. Such secret sharing schemes are called D'-out-of-D.

Shamir's D'-out-of-D secret sharing scheme operates over a finite field $\mathbf{Z}_p$, where $p > D$ is a prime sufficiently large so that all possible secrets may be represented in $\mathbf{Z}_p$. It has two procedures: Share and $\mathsf{Reconstruct}$:

• $\mathsf{Share}_{D',D}(x)$. The procedure samples a uniformly random polynomial $g(\cdot)$ over $\mathbf{Z}_p$, of degree at most $D'-1$, where the free coefficient is the secret s. That is, $g(x) = s + a_1 x + a_2 x^2 + \ldots + a_{D'-1} x^{D'-1}$, where a_j, $1 \leq j \leq D'-1$, are selected independently and uniformly at random from $\mathbf{Z}_p$. The procedure outputs D values, $s_d = g(d)$, $d \in [D]$, where s_d is the share given to the dth party, T_d, $d \in [D]$.

• $\mathsf{Reconstruct}_{D'}(s_1, \ldots, s_D)$. The procedure is given any selection of D' shares out of $\{s_1, \ldots, s_D\}$; it then interpolates a polynomial $g(\cdot)$ of degree at most $D'-1$ using the given point values, and it outputs $s = g(0)$. Clearly, any selection of D' shares out of the D shares will yield the same polynomial $g(\cdot)$ that was used to generate the shares, as D' point values determine a unique polynomial of degree at most $D'-1$. Hence, any selection of D' shares will issue the secret s. On the other hand, any selection of $D'-1$ shares

reveals nothing about the secret s (in the sense that every value in the field remains equally probable to be the secret s).

2.2. Distributed tallying and secure computations over secret shares

Our protocol involves a set of parties, $\{T_1, \ldots, T_D\}$, that are called *talliers*. In the protocol, each voter creates shares of his[2] private ballot and distributes them to the D talliers. Since those ballots are matrices over $\mathbf{Z}_p$, as we discuss in Section 3, the secret sharing is carried out for each matrix entry independently.

We will use Shamir's D'-out-of-D secret sharing with $D' := \lfloor (D+1)/2 \rfloor$ (namely, the value $(D+1)/2$ rounded down to the nearest integer). With that setting, at least half of the talliers would need to collaborate in order to recover the secret ballots. If the set of talliers is trusted to have an *honest majority*, then such a betrayal scenario is impossible. Namely, if more than half of the talliers are honest, in the sense that they would not attempt cheating, then even if all other dishonest talliers collude in attempt to recover the secret ballots from the shares that they hold, they will not be able to extract any information on the ballots. Higher values of D (and consequently of $D' := \lfloor (D+1)/2 \rfloor$) will imply greater security against coalitions of corrupted talliers, but they will also imply higher costs.

The first task of the talliers upon receiving the shares of a voter's ballot matrix is to verify that those shares correspond to a legal ballot (see Section 3.2 where we explain what constitutes a legal ballot). Then, at the end of the election period, the talliers need to compute the identity of the winning candidates, as dictated by the rule. These tasks would be easy if the talliers could use their shares in order to recover the ballots. However, they must not do so, in order to protect the voters' privacy. Instead, they must perform these computations on the distributed shares, without revealing the shares to each other. As we shall see later on, these computations boil down to the four specific tasks that we proceed to describe.

Let $x_1, \ldots, x_L$ be L secrets in the underlying field $\mathbf{Z}_p$ (L is any integer) and assume that the talliers hold D'-out-of-D shares of each of those secrets. Then the talliers can perform the following computational tasks *without* recovering the secrets $x_1, \ldots, x_L$ or learning anything about them:

[2]For the sake of simplicity, we keep referring to parties by the pronoun "he". In our context, those parties may be voters, who are humans of any gender, or talliers, that are typically (genderless) servers.

1. **Evaluating arithmetic functions.** If $y = f(x_1, \ldots, x_L)$, where f is some public arithmetic function, compute D'-out-of-D shares of y.

2. **Secure comparisons.** If $y = 1_{x_1 < x_2}$ is the bit that equals 1 if $x_1 < x_2$ and 0 otherwise[3], compute D'-out-of-D shares of y.

3. **Testing positivity.** If $p > 2N$ (where N is the number of voters) and $x_1 \in [-N, N]$, compute D'-out-of-D shares of the bit $1_{x_1 > 0}$.

4. **Testing equality to zero.** If $p > 2N$ and $x_1 \in [-N, N]$, compute D'-out-of-D shares of the bit $1_{x_1 = 0}$.

All those computational tasks can be carried out by secure multiparty computation (MPC) [50] sub-protocols. We treat those computations as black boxes. Namely, it is sufficient to understand what are their inputs and outputs, as described above, but there is no need to understand their internal operation. However, interested readers may refer to Appendix A for more information on those computations.

3. A secure order-based voting protocol

In this section, we describe our method for securely computing the winners in two order-based voting rules, Copeland and Maximin.[4] We begin with formal definitions in Section 3.1. Then, in Section 3.2, we characterize legal ballot matrices for each of the two rules. Such a characterization is an essential part of our method since the talliers need to verify, in an oblivious manner, that each cast ballot is indeed legal, and does not hide a malicious attempt to cheat or sabotage the elections. In other words, the talliers need to verify the legality of each cast ballot only through its secret shares, i.e., without getting access to the actual ballot. The characterization that we describe in Section 3.2 will be used later on to perform such an oblivious validation.

[3]For any predicate Π, we denote by 1_Π the bit that equals 1 if and only if (iff hereinafter) the predicate Π holds.

[4]Our protocols are designed for scenarios in which it is required to issue just the identity of the winners. In scenarios where more information is needed, such as a ranking of all candidates, or even a score for each candidate, our protocols can easily output also that additional information.

Then, in Section 3.3 we introduce our secure voting protocol. The protocol description includes sub-protocols for validating the cast ballots and a sub-protocol for computing the final election results from all legal ballots. The validation sub-protocols are described in Sections 3.4 and 3.5. The sub-protocols for computing the election results are described in Sections 3.6 and 3.7 for COPELAND and MAXIMIN rules, respectively.

We conclude this section with a discussion on how to set the size of the underlying field in which all computations take place (Section 3.8), and a discussion of the overall security of our protocol (Section 3.9).

3.1. Formal definitions

We consider a setting in which there are N voters, $\mathbf{V} = \{V_1, \ldots, V_N\}$, that wish to hold an election over M candidates, $\mathbf{C} = \{C_1, \ldots, C_M\}$. The output of the election is a ranking of $\mathbf{C}$. We proceed to define the two order-based rules for which we devise a secure MPC protocol in this section.

• COPELAND. Define for each V_n a matrix $P_n = (P_n(m, m') : m, m' \in [M])$, where $P_n(m, m') = 1$ if C_m is ranked higher than $C_{m'}$ in V_n's ranking, $P_n(m, m') = -1$ if C_m is ranked lower than $C_{m'}$, and all diagonal entries are 0. Then the sum matrix,

$$P = \sum_{n=1}^{N} P_n \,, \tag{1}$$

induces the following score for each candidate:

$$\mathbf{w}(m) := |\{m' \neq m : P(m, m') > 0\}| + \alpha |\{m' \neq m : P(m, m') = 0\}| \,. \tag{2}$$

Namely, $\mathbf{w}(m)$ equals the number of candidates $C_{m'}$ that a majority of the voters ranked lower than C_m, plus α times the number of candidates $C_{m'}$ who broke even with C_m. The parameter α can be set to any rational number between 0 and 1. The most common setting is $\alpha = \frac{1}{2}$; the COPELAND rule with this setting of α is known as COPELAND$^{\frac{1}{2}}$ [24].

• MAXIMIN. Define the matrices P_n so that $P_n(m, m') = 1$ if C_m is ranked higher than $C_{m'}$ in V_n's ranking, while $P_n(m, m') = 0$ otherwise. As in COPELAND rule, we let P denote the sum of all ballot matrices, see Eq. (1). Then $P(m, m')$ is the number of voters who preferred C_m over $C_{m'}$. The final score of C_m, $m \in [M]$, is then set to $\mathbf{w}(m) := \min_{m' \neq m} P(m, m')$.

In both these order-based rules, we shall refer hereinafter to the matrix P_n as V_n's *ballot matrix*, and to P as the aggregated ballot matrix.

3.2. Characterization of legal ballot matrices

Here, we characterize the ballot matrices in each of the two order-based rules that we consider. Such a characterization will be used later on in the secure voting protocol.

Theorem 1. *An $M \times M$ matrix Q is a valid ballot under the* Copeland *rule iff it satisfies the following conditions:*

1. $Q(m, m') \in \{-1, 1\}$ *for all* $1 \leq m < m' \leq M$;
2. $Q(m, m) = 0$ *for all* $m \in [M]$;
3. $Q(m', m) + Q(m, m') = 0$ *for all* $1 \leq m < m' \leq M$;
4. *If* $Q_m := \sum_{m' \in [M]} Q(m, m')$ *then* $Q_m \neq Q_{m'}$ *for all* $1 \leq m < m' \leq M$.

Theorem 2. *An $M \times M$ matrix Q is a valid ballot under the* Maximin *rule iff it satisfies the following conditions:*

1. $Q(m, m') \in \{0, 1\}$ *for all* $1 \leq m < m' \leq M$;
2. $Q(m, m) = 0$ *for all* $m \in [M]$;
3. $Q(m', m) + Q(m, m') = 1$ *for all* $1 \leq m < m' \leq M$;
4. *If* $Q_m := \sum_{m' \in [M]} Q(m, m')$ *then* $Q_m \neq Q_{m'}$ *for all* $1 \leq m < m' \leq M$.

Conditions 1-3 in both theorems are clear. As for condition 4, it states that the pairwise relations that Q defines are consistent with a linear ordering of the M candidates. As a counter-example, consider the following Copeland ballot matrix over $M = 3$ candidates:

$$Q = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}.$$

It complies with conditions 1-3 in Theorem 1, but violates condition 4 since $Q_1 = Q_2 = Q_3 = 0$. Hence, it is an illegal matrix. Indeed, the matrix states that $C_1 > C_2$ (since $Q(1, 2) = 1$), $C_2 > C_3$, and $C_3 > C_1$. Such cyclicity is impossible in a linear ordering.

Proof of Theorem 1. Assume that the ordering of a voter over the set of candidates $\mathbf{C}$ is $(C_{j_1}, \ldots, C_{j_M})$ where $\mathbf{j} := (j_1, \ldots, j_M)$ is some permutation of $[M]$. Then the ballot matrix that such an ordering induces is

$$Q = (Q(m, m'))_{1 \le m, m' \le M}$$

where $Q(m, m') = 1$ if m appears before m' in the sequence $\mathbf{j}$, $Q(m, m') = -1$ if m appears after m' in $\mathbf{j}$, and $Q(m, m) = 0$ for all $m \in [M]$. Such a matrix clearly satisfies conditions 1, 2 and 3 in the theorem. It also satisfies condition 4, as we proceed to show. Fix $m \in [M]$ and let $k \in [M]$ be the unique index for which $m = j_k$. Then the mth row in Q consists of exactly $k - 1$ entries that equal -1, $M - k$ entries that equal 1, and a single entry on the diagonal that equals 0. Hence, Q_m, which is the sum of entries in that row, equals $M + 1 - 2k$. Clearly, all those values are distinct, since the mapping $m \mapsto k$ is a bijection. That completes the first part of the proof: every legal COPELAND ballot matrix satisfies conditions 1-4.

Assume next that Q is an $M \times M$ matrix that satisfies conditions 1-4. Then for each $m \in [M]$, the mth row in the matrix consists of a single 0 entry on the diagonal where all other entries are either 1 or -1. Assume that the number of -1 entries in the row equals $k(m) - 1$, for some $k(m) \in [M]$, while the number of 1 entries equals $M - k(m)$. Then the sum Q_m of entries in that row equals $M + 1 - 2k(m)$. As, by condition 4, all Q_m values are distinct, then $k(m) \neq k(m')$ when $m \neq m'$. Stated otherwise, the sequence $\mathbf{k} := (k(1), \ldots, k(M))$ is a permutation of $[M]$. Let $\mathbf{j} := (j_1, \ldots, j_M)$ be the inverse permutation of $\mathbf{k}$; i.e., for each $m \in [M]$, $j_{k(m)} = m$. Then it is easy to see that the matrix Q is the COPELAND ballot matrix that corresponds to the ordering $\mathbf{j}$. That completes the second part of the proof: every matrix that satisfies conditions 1-4 is a legal COPELAND ballot matrix. □

The proof of Theorem 2 is similar to that of Theorem 1 and thus omitted.

We conclude this section with the following observation. Let us define a projection mapping $\Gamma : \mathbf{Z}_p^{M \times M} \mapsto \mathbf{Z}_p^{M(M-1)/2}$, which takes an $M \times M$ matrix $Q \in \mathbf{Z}_p^{M \times M}$ and outputs its upper triangle,

$$\Gamma(Q) := (Q(m, m') : 1 \le m < m' \le M)\,. \tag{3}$$

Conditions 2 and 3 in Theorems 1 and 2 imply that every ballot matrix, P_n, is fully determined by its upper triangle, $\Gamma(P_n)$, in either of the two voting rules that we consider.

3.3. The protocol

Here we present Protocol 1, a privacy-preserving implementation of the Copeland and Maximin order-based rules. The protocol computes, in a privacy-preserving manner, the winners in elections that are governed by those rules. It has two phases. We begin with a bird's-eye view of those two phases, and afterward, we provide a more detailed explanation.

Phase 1 (Lines 1-8) is the voting phase. In that phase, each voter V_n, $n \in [N] := \{1, \ldots, N\}$, constructs his ballot matrix, P_n (Line 2), and then creates and distributes to all talliers corresponding D'-out-of-D shares, with $D' = \lfloor (D+1)/2 \rfloor$, as described in Section 2.1 (Lines 3-7). Following that, the talliers jointly verify the legality of the shared ballot (Line 8). Phase 2 of the protocol (Lines 9-11) is carried out after the voting phase had ended. In that second phase, the talliers perform an MPC sub-protocol on the distributed shares in order to find the winning candidates, while remaining oblivious to the actual ballots.

After constructing his ballot matrix in Line 2, voter V_n, $n \in [N]$, samples a random share-generating polynomial of degree $D' - 1$ for each of the $M(M-1)/2$ entries in $\Gamma(P_n)$, where Γ is the projection mapping defined in Section 3.2 (Lines 3-5). Then, V_n sends each tallier T_d his relevant share of each of those entries, namely, the value of the corresponding share-generating polynomial at $x = d$, $d \in [D]$ (Lines 6-7).

In Line 8, the talliers engage in an MPC sub-protocol to verify the legality of V_n's ballot, without actually recovering that ballot. There are two validation sub-protocols that need to be executed and we describe them in Sections 3.4 and 3.5. Ballots that are found to be illegal are discarded. (In such cases it is possible to notify the voter that his ballot was rejected and allow him to resubmit it.)

Phase 2 (Lines 9-11) takes place at the end of the voting period, after the voters had cast their ballots. First (Lines 9-10), each of the talliers, T_d, $d \in [D]$, computes his D'-out-of-D share, denoted $G_d(m, m')$, in the (m, m')th entry of the aggregated ballot matrix P, see Eq. (1), for all $1 \leq m < m' \leq M$. This computation follows from the linearity of the secret-sharing scheme. Indeed, as $g_{n,m,m'}(d)$ is T_d's share of $P_n(m, m')$ in a D'-out-of-D Shamir's secret sharing, then $G_d(m, m') = \sum_{n \in [N]} g_{n,m,m'}(d)$ is a D'-out-of-D Shamir's secret share of $P(m, m') = \sum_{n \in [N]} P_n(m, m')$.

Then, in Line 11, the talliers engage in an MPC sub-protocol in order to find the indices of the K winning candidates. How can the talliers do that

when none of them actually holds the matrix P? We devote our discussion in Sections 3.6 and 3.7 to that interesting computational challenge.

Protocol 1: A basic protocol for secure order-based voting

Input: A set of M candidates $\mathbf{C}$; $K \in [M]$; a set of voters $\mathbf{V}$.

forall V_n, $n \in [N]$, **do**
  Construct the ballot matrix, P_n, according to the selected indexing of candidates and the voting rule
  forall $1 \leq m < m' \leq M$ **do**
    Select uniformly at random $a_{n,m,m',j} \in \mathbf{Z}_p$, $1 \leq j \leq D' - 1$
    Set $g_{n,m,m'}(x) = P_n(m, m') + \sum_{j=1}^{D'-1} a_{n,m,m',j} x^j$
  forall $d \in [D]$ **do**
    Send to T_d the set $\{(n, m, m', g_{n,m,m'}(d)) : 1 \leq m < m' \leq M\}$
  After all talliers receive their shares of V_n's ballot, they engage in an MPC sub-protocol to check its legality
forall T_d, $d \in [D]$ **do**
  Set $G_d(m, m') = \sum_{n \in [N]} g_{n,m,m'}(d)$, for all $1 \leq m < m' \leq M$
$T_1, \ldots, T_D$ find the indices of the K winners and publish them

Output: The K winning candidates from $\mathbf{C}$.

3.4. Verifying the legality of the cast ballots

Voters may attempt to cheat by submitting illegal ballots in order to help their preferred candidate. For example, a dishonest voter may send the talliers shares of the matrix cQ, where Q is his true ballot matrix and c is an "inflating factor" greater than 1. If such a cheating attempt remains undetected, then that voter would manage to multiply his vote by a factor of c. The shares of the illegal "inflated" matrix are indistinguishable from the shares of the legal ballot matrix (unless, of course, a majority of D' talliers collude and recover the ballot − a scenario that is impossible under our assumption that the talliers have an honest majority). Hence, it is necessary to devise a mechanism that would enable the talliers to check that the shares they received from each of the voters correspond to a legal ballot matrix, without actually recovering that matrix.

Malicious voters can sabotage the elections also in other manners. For example, a voter may create a legal ballot matrix Q and then alter the sign of some of the ± 1 entries in the shared upper triangle $\Gamma(Q)$, so that the resulting

matrix no longer corresponds to an ordering of the candidates. Even though such a manipulated matrix cannot serve a dishonest voter in an attempt to help a specific candidate, it still must be detected and discarded. Failing to detect the illegality of such a ballot may result in an aggregated matrix P that differs from the aggregated matrix P' that corresponds to the case in which that ballot is rejected. In such a case, the set of winners that P would determine may differ from that which P' determines. In summary, it is essential to validate each of the cast ballots in order to prevent any undesirable sabotaging of the elections.

In real-life electronic elections where voters typically cast their ballots on certified computers in voting centers, the chances of hacking such computers and tampering with the software that they run are small. However, for full-proof security and as a countermeasure against dishonest voters that might manage to hack the voting system, we proceed to describe an MPC solution that enables the talliers to verify the legality of each ballot, even though those ballots remain hidden from them. We note that in case a ballot is found to be illegal, the talliers may reconstruct it (by means of interpolation from the shares of D' talliers) and use the recovered ballot as proof of the voter's dishonesty. The ability to construct such proofs, which could be used in legal proceedings against dishonest voters, might deter voters from attempting cheating in the first place.

We proceed to explain how the talliers can verify the legality of the cast ballots in each of the two order-based rules. That validation is based on the characterizations of legal ballots as provided in Theorems 1 and 2 for Copeland and Maximin, respectively. Note that the talliers need only to verify conditions 1 and 4 (in either Theorem 1 or 2); condition 2 needs no verification since the voters do not distribute shares of the diagonal entries, as those entries are known to be zero; and condition 3 needs no verification since the voters distribute shares only in the upper triangle and then the talliers use condition 3 in order to infer the lower triangle from the shared upper triangle.

3.4.1. Verifying condition 1

Consider the shares that a voter distributed in $\Gamma(Q)$, where Q is his ballot matrix. The talliers need to verify that each entry in the shared $\Gamma(Q)$ is either 1 or -1 in Copeland, or either 1 or 0 in Maximin. The verification is performed independently on each of the $M(M-1)/2$ entries of the shared upper triangle. A shared scalar x is in $\{-1, 1\}$ (resp. in $\{0, 1\}$) iff

$(x+1)\cdot(x-1) = 0$ (resp. $x\cdot(x-1) = 0$). Hence, the talliers use their shares of x in order to compute the product $(x+1)\cdot(x-1)$ for COPELAND or $x\cdot(x-1)$ for MAXIMIN, using the procedure for evaluating arithmetic functions (as described in Section 2.2 and further discussed in Section Appendix A.1 in the Appendix). If the computed results are zero for all $M(M-1)/2$ entries of $\Gamma(Q)$, then $\Gamma(Q)$ satisfies condition 1 in Theorem 1 (COPELAND) or in Theorem 2 (MAXIMIN). If, on the other hand, at least one of the results is nonzero, then the ballot will be rejected.

3.4.2. Verifying condition 4

As the talliers received D'-out-of-D shares of each entry in $\Gamma(Q)$, they can compute D'-out-of-D shares of the corresponding row sums, Q_m, $m \in [M]$, as we proceed to show. In COPELAND, conditions 2 and 3 in Theorem 1 imply that

$$Q_m = \sum_{m' \in [M]} Q(m, m') = \sum_{m' > m} Q(m, m') - \sum_{m' < m} Q(m', m)\,. \tag{4}$$

Since the talliers hold shares of $Q(m', m)$ for all $1 \le m' < m \le M$, they can use Eq. (4) and the linearity of secret sharing to compute shares of Q_m, $m \in [M]$. In MAXIMIN, on the other hand, conditions 2 and 3 in Theorem 2 imply that

$$Q_m = \sum_{m' > m} Q(m, m') - \sum_{m' < m} Q(m', m) + (m-1)\,. \tag{5}$$

Here too, the linearity of secret sharing and the relation in Eq. (5) enable the talliers to compute shares of Q_m, $m \in [M]$, also in the case of MAXIMIN.

Now, it is necessary to verify that all M values Q_m, $m \in [M]$, are distinct. That condition can be verified by computing the product

$$F(Q) := \prod_{1 \le m' < m \le M} (Q_m - Q_{m'})\,. \tag{6}$$

Condition 4, in both rules, holds iff $F(Q) \neq 0$. Hence, the talliers, who hold shares of Q_m, $m \in [M]$, may compute $F(Q)$ and then accept the ballot iff $F(Q) \neq 0$.

3.4.3. Privacy

A natural question that arises is whether the above described validation process poses a risk to the privacy of the voters. In other words, a voter that

casts a legal ballot wants to be ascertained that the validation process only reveals that the ballot is legal, while all other information is kept hidden from the talliers. We proceed to examine that question.

The procedure for verifying condition 1 in Theorems 1 and 2 offers perfect privacy for honest voters. If the ballot Q is legal then all computed values will be zero. Hence, apart from the legality of the ballot, the talliers will not learn anything about the content of the ballot.

The procedure for verifying condition 4 in Theorems 1 and 2 offers *almost* perfect privacy, in the following sense. If Q is a valid ballot in COPELAND, then the ordered tuple $(Q_1, \ldots, Q_M)$ is a permutation of the ordered tuple $(-M+1, -M+3, \ldots, M-3, M-1)$. This statement follows from the proof of condition 4 in Theorem 1. Hence, as can be readily verified, if Q is a legal ballot then the value of $F(Q)$, Eq. (6), which the talliers compute in the validation procedure, equals

$$F(Q) = \pm 2^{\binom{M}{2}} \cdot \prod_{1 \le m' < m \le M} (m - m') \,, \tag{7}$$

where the sign of the product in Eq. (7) is determined by the signature of $(Q_1, \ldots, Q_M)$ when viewed as a permutation of the ordered tuple $(-M+1, -M+3, \ldots, M-3, M-1)$. Hence, since a ballot in COPELAND describes some ordering (permutation) of the candidates, the talliers will be able to infer the signature of that permutation, but nothing beyond that. To eliminate even that negligible leakage of information, the talliers will simply compute $F(Q)^2$ instead of $F(Q)$.

Similarly, the procedure for verifying condition 4 in Theorem 2, for MAXIMIN, is also privacy-preserving in the same manner. Indeed, in the case of MAXIMIN, $(Q_1, \ldots, Q_M)$ is a permutation of the ordered tuple $(0, 1, \ldots, M-2, M-1)$, and, therefore,

$$F(Q) = \pm \prod_{1 \le m' < m \le M} (m - m') \,,$$

where the sign of the above product is determined by the signature of $(Q_1, \ldots, Q_M)$ as a permutation of $(0, 1, \ldots, M-2, M-1)$. Here too, the talliers will recover $F(Q)^2$ instead of $F(Q)$ to ensure perfect privacy.

3.5. Verifying the legality of the secret sharing

A malicious voter V may attempt to sabotage the election by distributing to the D talliers shares that do not correspond to a polynomial of degree $D'-1$.

Namely, if $\{s_1, \ldots, s_D\}$ are the shares that V distributed to $T_1, \ldots, T_D$ in one of the entries in his ballot matrix, it is possible that there is no polynomial g of degree (up to) $D' - 1$ such that $g(d) = s_d$ for all $d \in [D]$. By carefully selecting those shares, they may still pass the verification tests as described in Section 3.4 with some non-negligible probability, and then they would be integrated in the final computation of the winners. Since such shares do not correspond to a legal vote, they may sabotage the final computation of the winners (as we describe later on in Sections 3.6 and 3.7). To prevent such an attack, we explain herein how the talliers may detect it without learning anything on the submitted ballot (beyond the mere legality of the secret sharing that was applied to it).

Sub-protocol 2 performs the desired testing. The talliers apply it on the secret shares $\{s_1, \ldots, s_D\}$ that they had received in any of the entries $s = Q(m', m)$ in an incoming ballot matrix Q.

First, each tallier T_d, $d \in [D]$, produces a random number r_d and distributes to all talliers D'-out-of-D shares of it (Lines 1-3). Then, each tallier T_d adds to his share, s_d, the shares received from all talliers in the random numbers that they had produced, and broadcasts the result, denoted $\hat{s}_d$ (Lines 4-6). The talliers proceed to find, by means of interpolation, a polynomial f that satisfies $f(d) = \hat{s}_d$ for all $d \in [D]$ (Line 7). If the degree of the polynomial is $D' - 1$ or less, the set of values $\{s_1, \ldots, s_D\}$ constitute a legal D'-out-of-D sharing in some (still unknown) secret $s \in \mathbf{Z}_p$; otherwise, it is not (Lines 8-11).

Lemma 3. *Sub-protocol 2 is correct and fully preserves the voter's privacy.*

Proof. Assume that V is honest. Then for each entry $s = Q(m', m)$ in his ballot matrix there exists a polynomial g of degree at most $D' - 1$ such that $s_d = g(d)$, $d \in [D]$. Define

$$G := g + \sum_{j \in [D]} g_j \,, \tag{8}$$

where g_j, $j \in [D]$, is the polynomial of degree $D'-1$ that T_j used for generating the secret shares of its random value r_j (Line 3 in Sub-protocol 2). Then

$$G(d) = g(d) + \sum_{j \in [D]} g_j(d) = s_d + \sum_{j \in [D]} r_{j,d} = \hat{s}_d \,, \qquad d \in [D] \,. \tag{9}$$

Hence, the interpolating polynomial f that the talliers compute in Line 7 coincides with G. As G is a sum of polynomials of degree $D' - 1$ at most, the

Sub-Protocol 2: Verifying that the shares distributed in some scalar secret constitute a legal D'-out-of-D sharing

Input: T_d, $d \in [D]$, has s_d, a supposedly D'-out-of-D share in some secret $s \in \mathbf{Z}_p$.

1 **forall** $d \in [D]$ **do**
2 | T_d produces a random $r_d \in \mathbf{Z}_p$
3 | T_d distributes to all talliers D'-out-of-D shares of r_d, denoted $r_{d,1}, \ldots, r_{d,D}$
4 **forall** $d \in [D]$ **do**
5 | T_d computes $\hat{s}_d = s_d + \sum_{j \in [D]} r_{j,d}$
6 | T_d broadcasts $\hat{s}_d$
7 All talliers compute a polynomial f of degree up to $D-1$ such that $f(d) = \hat{s}_d$, $d \in [D]$
8 **if** $\deg f \leq D' - 1$ **then**
9 | **Output** "A legal D'-out-of-D sharing"
10 **else**
11 | **Output** "An illegal D'-out-of-D sharing"

Output : The legality of the secret sharing operation.

validation in Lines 8-9 will pass successfully. In that case, the talliers would learn $G(0) = g(0) + \sum_{j \in [D]} r_j$. Here, $g(0)$ is the secret entry in V's ballot matrix and r_j is the secret random value that T_j had chosen, $j \in [D]$. Hence, $G(0)$ reveals no information on $g(0)$. Since all other coefficients in G are also random numbers that are not related to $g(0)$, this validation procedure provides perfect privacy for the voter.

Assume next that V had distributed illegal shares for one of his ballot matrix entries. Namely, V had distributed shares $\{s_1, \ldots, s_D\}$ such that the minimum-degree polynomial g that satisfies $g(d) = s_d$ for all $d \in [D]$ is of degree $t > D' - 1$. In that case, the polynomial f that the talliers compute in Line 7 would be $f = g + \sum_{j \in [D]} g_j$. Since the degree of that polynomial is $t > D' - 1$, they would reject the ballot (Lines 10-11). Hence, the verification procedure is correct and provides perfect privacy. □

When the talliers receive shares in some ballot matrix, they must first execute Sub-protocol 2 for each of its entries, and only if that validation passes successfully they will proceed to perform the validation described in Section 3.4.

Finally, we note that Lines 1-3 in Sub-protocol 2 can be executed even before the election period starts. Namely, once the number of registered voters, N, and the number of candidates, M, are determined, the talliers can produce $NM(M-1)/2$ random values and distribute shares of them to be used later on in masking the $M(M-1)/2$ entries in each voter's ballot matrix.

3.6. An MPC computation of the winners in the COPELAND *rule*

The parameter α in Eq. (2) is always a rational number; typical settings of α are 0, 1, or $\frac{1}{2}$ [24]. Assume that $\alpha = \frac{s}{t}$ for some integers s and t. Then

$$t \cdot \mathbf{w}(m) = t \cdot \sum_{m' \in [M] \setminus \{m\}} 1_{P(m,m')>0} + s \cdot \sum_{m' \in [M] \setminus \{m\}} 1_{P(m,m')=0} \,. \tag{10}$$

The expression in Eq. (10) involves all entries in P outside the diagonal. However, the talliers hold D'-out-of-D shares, denoted $G_d(m,m')$, $d \in [D]$, in $P(m,m')$ only for entries above the diagonal, $1 \leq m < m' \leq M$ (see Lines 9-10 in Protocol 1). Hence, we first translate Eq. (10) into an equivalent expression that involves only entries in P above the diagonal. Condition 3 in Theorem 1, together with Eq. (1), imply that $P(m',m) = -P(m,m')$. Hence, for all $m' < m$, we can replace $1_{P(m,m')=0}$ with $1_{P(m',m)=0}$, while $1_{P(m,m')>0}$ can be replaced with $1_{-P(m',m)>0}$. Hence,

$$\begin{aligned} t \cdot \mathbf{w}(m) = \\ & t \cdot \left\{ \sum_{m'>m} 1_{P(m,m')>0} + \sum_{m'<m} 1_{-P(m',m)>0} \right\} + \\ & s \cdot \left\{ \sum_{m'>m} 1_{P(m,m')=0} + \sum_{m'<m} 1_{P(m',m)=0} \right\} \end{aligned} \tag{11}$$

Eq. (11) expresses the score of candidate C_m, re-scaled by a factor of t, only by entries in P above the diagonal, in which the talliers hold D'-out-of-D secret shares.

In view of the above, the talliers may begin by computing secret shares of the bits of positivity in the first sum on the right-hand side of Eq. (11), by using the MPC sub-protocol described in Section Appendix A.3. As for the bits of equality to zero in the second sum on the right-hand side of Eq. (11), the talliers can compute secret shares of them using the MPC sub-protocol

described in Section Appendix A.4. As the value of $t \cdot \mathbf{w}(m)$ is a linear combination of those bits, the talliers can then use the secret shares of those bits and Eq. (11) in order to get secret shares of $t \cdot \mathbf{w}(m)$, for each of the candidates, C_m, $m \in [M]$.

Next, they perform secure comparisons among the values $t\mathbf{w}(m)$, $m \in [M]$, in order to find the K candidates with the highest scores. To do that, they need to perform $M-1$ secure comparisons (as described in Section 2.2 and then further discussed in Section Appendix A.2) in order to find the candidate with the highest score, $M-2$ additional comparisons to find the next one, and so forth down to $M-K$ comparisons in order to find the Kth winning candidate. Namely, the overall number of comparisons in this final stage is

$$\sum_{m=M-K}^{M-1} m = K \cdot \left(M - \frac{K+1}{2} \right) .$$

The above number is bounded by $M(M-1)/2$ for all $K < M$. Once this computational task is concluded, the talliers publish the indices of the K winners (Line 11 in Protocol 1)).

We summarize the above described computation in Sub-protocol 3, which is an implementation of Line 11 in Protocol 1. It assumes that the talliers hold D'-out-of-D secret shares of $P(m, m')$ for all $1 \leq m < m' \leq M$. Indeed, that computation has already taken place in Lines 9-10 of Protocol 1. Sub-protocol 3 starts with a computation of D'-out-of-D shares of all of the positivity bits and equality to zero bits that relate to the entries above the diagonal in P (Lines 1-4). Then, in Lines 5-7, the talliers use those shares in order to obtain D'-out-of-D shares of $t \cdot \mathbf{w}(m)$ for each of the candidates, using Eq. (11); the D'-out-of-D shares of $t \cdot \mathbf{w}(m)$ are denoted $\{w_d(m) : d \in [D]\}$. Finally, using the secure comparison sub-protocol, they find the K winners (Lines 8-10).

3.6.1. Privacy

The talliers hold D'-out-of-D shares of each of the ballot matrices, P_n, $n \in [N]$, as well as in the aggregated ballot matrix P. Under our assumption of honest majority, and our setting of $D' = \lfloor (D+1)/2 \rfloor$, the talliers cannot recover any entry in any of the ballot matrices nor in the aggregated ballot matrix. In computing the final election results, they input the shares that they hold into the secure comparison sub-protocol, the positivity testing sub-protocol, or the sub-protocol that tests equality to zero (Sections Appendix A.2–Appendix A.4). The secure comparison sub-protocol is perfectly

Sub-Protocol 3: Determining the winners in COPELAND rule

Input: T_d, $d \in [D]$, has $G_d(m,m')$ (a share of $P(m,m')$) for all $1 \le m < m' \le M$.

forall $1 \le m < m' \le M$ **do**
- The talliers apply the positivity sub-protocol to translate $\{G_d(m,m') : d \in [D]\}$ into shares $\{\sigma_d^+(m,m') : d \in [D]\}$ in $1_{P(m,m')>0}$;
- The talliers apply the positivity sub-protocol to translate $\{G_d(m,m') : d \in [D]\}$ into shares $\{\sigma_d^-(m,m') : d \in [D]\}$ in $1_{-P(m,m')>0}$;
- The talliers apply the equality to zero sub-protocol to translate $\{G_d(m,m') : d \in [D]\}$ into shares $\{\sigma_d^0(m,m') : d \in [D]\}$ in $1_{P(m,m')=0}$;

forall $d \in [D]$ **do**
- **forall** $m \in [M]$ **do**
  - T_d computes $w_d(m) = t \cdot \left\{\sum_{m'>m} \sigma_d^+(m,m') + \sum_{m'<m} \sigma_d^-(m',m)\right\} + s \cdot \left\{\sum_{m'>m} \sigma_d^0(m,m') + \sum_{m'<m} \sigma_d^0(m',m)\right\}$;

forall $k \in [K] := \{1,\ldots,K\}$ **do**
- The talliers perform $M-k$ invocations of the secure comparison sub-protocol over the $M-k+1$ candidates in $\mathbf{C}$ in order to find the kth elected candidate;
- The talliers output the candidate that was found and remove him from $\mathbf{C}$;

Output: The K winning candidates from $\mathbf{C}$.

secure, as was shown in [39]. The positivity testing sub-protocol that we presented here is just an implementation of one component from the secure comparison sub-protocol, hence it is also perfectly secure. Finally, the testing of equality to zero invokes the protocol for evaluating arithmetic functions of [19] and [14], which was shown there to be secure. Hence, Sub-protocol 3 is perfectly secure.

3.7. An MPC computation of the winners in the MAXIMIN *rule*

Fixing $m \in [M]$, the talliers need first to find the index $m' \neq m$ which minimizes $P(m,m')$ (where P is, as before, the sum of all ballot matrices, see Eq. (1)). Once m' is found then $\mathbf{w}(m) = P(m,m')$. To do that (finding a

minimum among $M-1$ values), the talliers need to perform $M-2$ secure comparisons. That means an overall number of $M(M-2)$ secure comparisons for the first stage in the talliers' computation of the final results (namely, the computation of the scores for all candidates under the MAXIMIN rule). The second stage is just like in COPELAND, namely, finding the indices of the K candidates with the highest $\mathbf{w}$ scores. As analyzed earlier, that task requires an invocation of the secure comparison sub-protocol at most $M(M-1)/2$ times. Namely, the determination of the winners in the case of MAXIMIN requires performing the comparison sub-protocol less than $1.5M^2$ times.

The above described computation maintains the privacy of the voters, as argued in Section 3.6.1.

3.8. A lower bound on the field's size

Here we comment on the requirements of our protocol regarding the size p of the underlying field $\mathbf{Z}_p$.

The prime p should be selected to be greater than the following four values:

(i) D, as that is the number of talliers (see Section 2.1).

(ii) $2N$, since the field should be large enough to hold the entries of the sum P of all ballot matrices, Eq. (1), and the entries of that matrix are confined to the range $[-N, N]$.

(iii) $\max\{t, s\} \cdot (M-1)$, since that is the upper bound on $t \cdot \mathbf{w}(m)$, see Eq. (10), which is secret-shared among the talliers.

(iv) $2(M-1)$, since in validating a given ballot matrix Q, the talliers need to test the equality to zero of $F(Q)$, see Eq. (6). As $F(Q)$ is a product of the differences $Q_m - Q_{m'}$, and each of those differences can be at most $2(M-1)$ (in COPELAND) or $M-1$ (in MAXIMIN), it is necessary to set p to be larger than that maximal value.

Hence, in summary, p should be selected to be larger than each of the above four values. Since D (number of talliers), M (number of candidates), and s and t (the numerator and denominator in the coefficient α in COPELAND rule, Eq. (2)), are typically much smaller than N, the number of voters, the essential lower bound on p is $2N$. In our evaluation, we selected $p = 2^{31} - 1$, which is sufficiently large for any conceivable election scenario.

3.9. The security of the protocol

An important goal of secure voting is to provide anonymity; namely, it should be impossible to connect a ballot to the voter who cast it. Protocol 1

achieves that goal, and beyond. Indeed, each cast ballot is distributed into D'-out-of-D random shares and then each share is designated to a unique tallier. Under the *honest majority* assumption, the voters' privacy is perfectly preserved. Namely, unless at least $D' \geq D/2$ talliers betray the trust vested in them and collude, the ballots remain secret. Therefore, not only that a ballot cannot be connected to a voter, even the content of the ballots is never exposed, and not even the aggregated ballot matrix. The only information that anyone learns at the end of Protocol 1's execution is the final election results. This is a level of privacy that exceeds mere anonymity.

A scenario in which at least half of the talliers collude is highly improbable, and its probability decreases as D increases. Ideally, the talliers would be parties that enjoy a high level of trust within the organization or state in which the elections take place, and whose business is based on such trust. Betraying that trust may incur devastating consequences for the talliers. Hence, even if D is set to a low value such as $D = 5$ or even $D = 3$, the probability of a privacy breach (namely, that $D' = \lfloor (D+1)/2 \rfloor$ talliers choose to betray the trust vested in them) would be small.

Instead of using a D'-out-of-D threshold secret sharing, we could use an additive "All or Nothing" secret sharing scheme, in which **all** D shares of all D talliers are needed in order to reconstruct the shared secrets and use suitable MPC protocols for multiplication and comparison. The necessary cryptographic machinery for such D-out-of-D secret sharing was already developed in [4], so our protocols can be readily converted to such a setting. That approach would result in higher security since the privacy of the voters would be jeopardized only if all D talliers betray the trust vested in them and not just D' out of them. However, such a scheme is not robust: if even a single tallier is attacked or becomes dysfunctional for whatever reason, then all ballots would be lost. Such a risk is utterly unacceptable in voting systems. Our protocols, on the other hand, can withstand the loss of $D - D'$ talliers.

In view of the above discussion, the tradeoff in setting the number of talliers D is clear: higher values of D provide higher security since more talliers would need to be corrupted in order to breach the system's security. Higher values of D also provide higher robustness, since the system can withstand higher numbers of tallier failures. However, increasing D has its costs: more independent and reputable talliers are needed, and the computational costs of our protocols increase, albeit modestly (see Section 5.4).

We would like to stress that our protocols focus on securing the computation of the election results. Needless to say that those protocols must be

integrated into a comprehensive system that takes care of other aspects of voting systems. For example, it is essential to guarantee that only registered voters can vote and that each one can vote just once. It is possible to ensure such conditions by standard means. Another requirement is the need to prevent attacks of malicious adversaries. In the context of our protocols, an adversary may eavesdrop on the communication link between some voter V_n and at least D' of the talliers, and intercept the messages that V_n sends to them (in Protocol 1's Line 7) in order to recover V_n's ballot. That adversary may also replace V_n's original messages to all D talliers with other messages (say, ones that carry shares of a ballot that reflects the adversary's preferences). Such attacks can be easily thwarted by requiring each party (a voter or a tallier) to have a certified public key, encrypt each message that he sends out using the receiver's public key and then sign it using his own private key; also, when receiving messages, each party must first verify them using the public key of the sender and then send a suitable message of confirmation to the sender. Namely, each message that a voter V_n sends to a tallier T_d in Line 7 of Protocol 1 should be signed with V_n's private key and then encrypted by T_d's public key; and T_d must acknowledge its receipt and verification.

4. Order-based voting with ties

So far we concentrated on the case in which the voters submit a strict ordering of the candidates, see Section 3.1. However, order-based rules allow voters to set ties between candidates. For example, if there are six candidates, $C_1, \ldots, C_6$, a voter may identify C_1 as her most preferred candidate, C_2, C_3, C_4 as her second-tier candidates, where she is indifferent between them, and C_5, C_6 as her least preferred candidates. In that case, her ballot matrix would be

$$Q = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 & 0 \end{pmatrix}. \tag{12}$$

The aggregation of ballots and the determination of the election results is carried out as described earlier: the score $\mathbf{w}(m)$ for each candidate C_m is computed by Eqs. (1)+(2), and then the winners are those with the highest scores.

The challenge that allowing ties brings about is in the characterization of legal ballot matrices and in the design of an MPC protocol for securely verifying the legality of a voter's shared ballot matrix (in the sense that it corresponds to an ordering of the candidates with possible ties). That is what we do in Sections 4.1 and 4.2 below. Our focus in those sections is the COPELAND rule. The MAXIMIN rule with ties is discussed in Section 4.3.

4.1. Characterizing legal ballot matrices when ties are allowed

Theorem 4 characterizes legal ballot matrices in the COPELAND rule when ties are allowed. It is a generalization of Theorem 1.

Theorem 4. *An $M \times M$ matrix Q is a valid ballot under the* COPELAND *rule with ties iff it satisfies the following conditions:*

1. $Q(m, m') \in \{-1, 0, 1\}$ *for all* $1 \leq m < m' \leq M$;
2. $Q(m, m) = 0$ *for all* $m \in [M]$;
3. $Q(m', m) + Q(m, m') = 0$ *for all* $1 \leq m < m' \leq M$;
4. *If* $Q(m', m) = 0$ *then* $Q(m, \cdot) = Q(m', \cdot)$.
5. *For all* $m \in [M]$ *set* $\eta_m = 1$ *if* $Q(m', m) \neq 0$ *for all* $m' < m$, *and* $\eta_m = 0$ *otherwise. Define* $Q_m := \sum_{m' \in [M]} \eta_{m'} \cdot Q(m, m')$. *Then* $Q_{m'} \neq Q_m$ *for all* $1 \leq m' < m \leq M$ *such that* $\eta_{m'} = \eta_m = 1$.

The differences between Theorem 1 (no ties) and Theorem 4 (ties allowed) are in the first condition (where in the latter theorem zero entries are allowed also in non-diagonal entries), condition 4 (that holds trivially in Theorem 1, as there are no zeroes outside the diagonal), and in condition 5 that generalizes condition 4 in Theorem 1 (because when there are no ties we have $\eta_m = 1$ for all $m \in [M]$).

Before proving the theorem, we discuss conditions 4 and 5 and exemplify their necessity. The rational behind condition 4 is as follows. If $Q(m, m') = 0$ then candidates C_m and $C_{m'}$ are equivalent for the voter, and hence their relative ranking with respect to each of the other candidates (as manifested by their corresponding rows in the matrix) must be the same. As for condition 5, let us first define the relation $\sim$ on $[M]$ as follows: $m \sim m'$ iff C_m and $C_{m'}$ are in a tie (namely, $Q(m', m) = 0$). That is obviously an equivalence relation. Let $[M]/\sim$ denote the quotient set, and $k := |[M]/\sim|$ be the number of

equivalence classes, i.e., the number of distinct tiers in the ranking. In what follows we represent each equivalence class by the minimal index in it. Then condition 5 considers the reduced $k \times k$ matrix over the set of representative indices, and it restates condition 4 from Theorem 1, namely, that all row sums in that reduced (and tie-free) matrix must be distinct. For the sake of illustration, the ballot matrix Q in Eq. (12) induces $k = 3$ equivalence classes, $[M]/\sim = \{\{1\}, \{2, 3, 4\}, \{5, 6\}\}$. The representative indices are $1, 2, 5$ and the reduced 3×3 matrix is the sub-matrix of Q that consists of its 1st, 2nd, and 5th rows and columns,

$$Q|_{1,2,5} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} .$$

Indeed, the row sums in that reduced matrix are distinct.

Next, we give examples to matrices that comply with conditions 1-3 of Theorem 4, but violate condition 4 or 5:

$$Q_1 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & -1 \\ -1 & -1 & 1 & 0 \end{pmatrix} , \quad Q_2 = \begin{pmatrix} 0 & 1 & -1 & -1 \\ -1 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix} .$$

Q_1 violates condition 4: on one hand, $Q_1(2, 3) = Q_1(3, 2) = 0$, which means that C_2 and C_3 are in a tie, but the corresponding rows differ, since $Q_1(2, 4) = 1$ while $Q_1(3, 4) = -1$. However, there can be no ranking of the candidates in which C_4 is ranked lower than C_2 and higher than C_3, while at the same time C_2 and C_3 are in a tie. As for Q_2, it does comply with condition 4: indeed, there are two candidates in a tie, C_3 and C_4, and their corresponding rows are equal. But Q_2 violates condition 5: indeed, while $\eta_1 = \eta_2 = \eta_3 = 1$ (and $\eta_4 = 0$) we have $Q_1 = Q_2 = Q_3 = 0$, in contradiction with condition 5 that requires those row sums to be distinct. That equality of the row sums is the result of the circular (and hence illegal) ranking between the candidates: $C_1 > C_2 > C_3 = C_4 > C_1$.

We now proceed to prove Theorem 4.

Proof. Assume that Q is a legal COPELAND ballot matrix with ties; namely, it is a matrix that describes some given ranking over $[M]$. Then it clearly satisfies conditions 1-3. As for condition 4, it merely states that two candidates

that are in a tie ($Q(m', m) = 0$) must have the same relative ranking vis-a-vis all candidates, hence it clearly holds. Finally, the row sums Q_m in condition 5 are the row sums in the reduced ballot matrix over $[M]/\sim$, where $\sim$ is as defined earlier ($m \sim m'$ if C_m and $C_{m'}$ are in a tie in the underlying ranking). Since there are no ties over $[M]/\sim$, those row sums must be distinct as implied by Theorem 1.

Assume now that Q is an $M \times M$ matrix that complies with conditions 1-5. We proceed to show that it induces a ranking over $[M]$. First, we say that $m \sim m'$, for $m, m' \in [M]$, if $Q(m', m) = 0$. Clearly, that relation is reflexive (condition 2), symmetric (condition 3), and transitive (as implied by condition 4). Hence, it is an equivalence. Let us represent each equivalence class by the minimal index in it, and let $[M]/\sim$ denote the set of representative indices and $k := |[M]/\sim|$ denote their number. Then $\eta_m = 1$ iff $m \in [M]/\sim$. Let Q' be the reduced $k \times k$ matrix that is obtained from Q by retaining only rows and columns of indices $m \in [M]/\sim$. Then Q' is a matrix that complies with all four conditions in Theorem 1 (where condition 4 in Theorem 1 follows from condition 5 herein). Hence, Q' induces a ranking (with no ties) over $[M]/\sim$. It follows that Q induces a ranking with ties over $[M]$ by adding each index in $[M] \setminus ([M]/\sim)$ to the ranking over $[M]/\sim$, next to the index in $[M]/\sim$ that is equivalent to it, with a tie between them. The resulting ranking with ties over $[M]$ is consistent with Q. □

4.2. Verifying the legality of the cast ballots when ties are allowed

Recall that if a voter's ballot matrix is Q then it suffices to distribute shares only in its upper triangle $\Gamma(Q)$, Eq. (3), see Section 3.2. Herein we explain how to verify the conditions of Theorem 4. As before, conditions 2 and 3 do not need to be verified since the talliers complete $\Gamma(Q)$ into a full ballot matrix Q in accordance with those anti-symmetry conditions.

The talliers can verify condition 1 if for all $1 \leq m' < m \leq M$ they compute shares of

$$x := Q(m', m) \cdot (Q(m', m) + 1) \cdot (Q(m', m) - 1)\,,$$

using the techniques described in Section Appendix A.1, recover x and verify that $x = 0$. If all those $M(M-1)/2$ tests pass successfully then Q complies with condition 1 and no other information on Q is revealed.

For the verification of conditions 4 and 5 the talliers compute for each $Q(m', m)$, $1 \leq m' < m \leq M$, shares of $\xi_{m',m} := 1_{Q(m',m)=0}$, where the latter computation is carried out as described in Section Appendix A.4. After that

preliminary computation, the talliers compute shares of

$$\pi_{m',m,k} := \xi_{m',m} \cdot (Q(m',k) - Q(m,k)) \ , \quad 1 \leq m' < m \leq M \ , \quad k \in [M] \ ,$$

as described in Section Appendix A.1, and then recover $\pi_{m',m,k}$. Condition 4 is verified iff

$$\pi_{m',m,k} = 0 \ , \quad 1 \leq m' < m \leq M \ , \quad k \in [M] \ . \tag{13}$$

Indeed, $\pi_{m',m,k} = 0$ iff either $\xi_{m',m} = 0$ or $Q(m',k) = Q(m,k)$ for all $k \in [M]$. Since $\xi_{m',m} = 0$ iff $Q(m',m) \neq 0$, Eq. (13) is equivalent to condition 4 in Theorem 4. Hence, a matrix Q passes that test iff it complies with condition 4, and if it does, no further information on it is leaked as a result of that verification. Note that the talliers do not hold shares of $Q(k,m)$ whenever $k \geq m$. Hence, in such cases the talliers rely on conditions 2 and 3 and make the following substitutions: when $k = m$ they set $Q(k,m) = 0$, and when $k > m$ they set $Q(k,m) = -Q(m,k)$.

Finally, we turn to the verification of condition 5. Here, the talliers compute for all $m \in [M]$ shares of

$$\eta_m := \prod_{m'=1}^{m-1} (1 - \xi_{m',m}) \ . \tag{14}$$

It can be easily verified that $\eta_m = 1$ if $Q(m',m) \neq 0$ for all $1 \leq m' < m$, while $\eta_m = 0$ otherwise. Hence, the indicator variables η_m as defined above equal the indicator variables η_m that are defined in Theorem 4's condition 5. The computation in Eq. (14) is carried out by the method described in Section Appendix A.1. Next, the talliers compute shares of

$$Q_m = \sum_{m' \in [M]} \eta_{m'} \cdot Q(m,m') \ , \tag{15}$$

where, as before, we recall that $Q(m,m) = 0$ and when $m' > m$ we rely on the equality $Q(m',m) = -Q(m,m')$. Condition 5 requires that for all $1 \leq m' < m \leq M$, whenever $\eta_m \eta_{m'} \neq 0$ then also $Q_{m'} - Q_m \neq 0$. That condition can be rephrased as follows:

$$\gamma_{m',m} := (1 - \eta_m \eta_{m'}) + \eta_m \eta_{m'} \cdot (Q_{m'} - Q_m) \neq 0 \ , \quad 1 \leq m' < m \leq M \ . \tag{16}$$

Indeed, if $\eta_m = 0$ or $\eta_{m'} = 0$, a case in which condition 5 is void, then the first addend in Eq. (16) equals 1 while the second addend is zero, so $\gamma_{m',m} = 1$.

If, on the other hand, $\eta_m \eta_{m'} = 1$, then condition 5 requires the difference $Q_{m'} - Q_m$ to be nonzero, and since in that case $\gamma_{m',m} = Q_{m'} - Q_m$, the inequalities in Eq. (16) are indeed equivalent to condition 5.

The talliers could compute shares of $\gamma_{m',m}$, using the techniques described in Section Appendix A.1, and then recover $\gamma_{m',m}$ and check that it is nonzero. Alas, the value of $\gamma_{m',m}$ will reveal information on the ballot matrix. To refrain from such leakage of information, the talliers produce offline (even prior to the election period) a large pool of shared nonzero random values from the underlying field $\mathbf{Z}_p$. Then, for each value $\gamma_{m',m}$ they would pull a fresh nonzero random value r from that pool and then recover $r\gamma_{m',m}$, using the shares that they had computed in $\gamma_{m',m}$, the shares that they had computed offline in r, and the multiplication procedure from Section Appendix A.1. Since $r\gamma_{m',m} \neq 0$ iff $\gamma_{m',m} \neq 0$, the talliers can verify condition 5 without learning any further information on Q.

We summarize the verification procedure in Sub-protocol 4. In Lines 1-5 the talliers complete $\Gamma(Q)$ into a full ballot matrix Q, based on Q's asymmetry. In Lines 6-10 they verify condition 1; in Lines 11-17 they verify condition 4; and in Lines 18-29 they verify condition 5. In any event of detecting that the ballot is illegal, the sub-protocol outputs a message about that illegality and aborts (Lines 10, 17, 29). If the sub-protocol reaches the end, it outputs a message about the legality of the input ballot matrix and then terminates (Line 30). Note that the computation in Line 25 is independent of the input and can be carried out offline, even before the election period had started.

4.3. The Maximin *rule with ties*

As discussed in Section 3.1, the ballot matrix in Maximin consists of 0,1 entries, where the (m, m')-th entry in the matrix equals 1 iff C_m is ranked strictly higher than $C_{m'}$ in the voter's ranking. Instead of formalizing a characterization and designing a corresponding verification MPC protocol for such matrices, we propose that also in Maximin rule, just like in Copeland, the voters will submit a matrix of $\{-1, 0, 1\}$-entries that fully spells out their ranking. The talliers can then verify the legality of that matrix by running Sub-protocol 4. After the ballot matrix is verified to be legal, the talliers may easily translate it to a Maximin ballot matrix of $\{0, 1\}$-entries, as described in Section 3.1, by running the following computation on each of the matrix entries, $x = Q(m, m')$, $1 \leq m \neq m' \leq M$:

$$x \leftarrow x + 1_{x+1=0}\,. \tag{17}$$

The computation in Eq. (17) involves an invocation of the equality to zero protocol of Section Appendix A.4. It is easy to see that the computation in Eq. (17) substitutes (shares of) $x = -1$ with (shares of) $x = 0$, but leaves (the shares of) x unchanged if $x \neq -1$. After that update, the ballot matrix (that was already verified to be a legal COPELAND ballot matrix) will be a legal MAXIMIN matrix, consisting only of $0, 1$ entries, and the aggregation of all ballots and the computation of the election results will be as described in Section 3.7.

5. An implementation of a secure order-based voting system

Our goal herein is to establish the practicality of our protocol and demonstrate it in action. To that end, we had implemented a demo of a voting system based on the protocols that we presented here for the COPELAND and MAXIMIN voting rules. The full source code of the demo is open source and is available on GitHub (https://github.com/arthurzam/SecureVoting). The demo is available at https://securevoting.ddns.net/. Interested readers can use it to define a new election (by setting all the necessary parameters) and invite voters to cast their votes and then compute the final election results.

The system is implemented in Python. We chose that programming language for ease of development and best cross-platform support. In addition, it makes the code easier to read, verify, and modify as required.

In our system there are users (clients) and talliers (servers). A user can be either an election manager, namely a party that initiates a new election and defines its characteristics, or a voter in an election that was initiated and is managed by another user. As for the talliers, even though our code can be executed with any number of talliers, we focus in our demo on the case of three talliers.

The demo uses docker and docker-compose to easily deploy on servers. The deployment includes a nearly-static Web server (which supplies the user interface), a PostgreSQL database (holding details on all currently defined elections, either future ones, on-going elections, or complete elections, and on all voters), and the tallier module. A high-level block diagram of the demo is given in Figure 2. For the sake of simplicity, the figure illustrates only two talliers; the structure of all talliers, and the communication between each tallier and the client and between each pair of talliers is as shown in the figure for Talliers 1 and 2.

Sub-Protocol 4: Verifying the legality of a ballot matrix in COPELAND rule with ties

Input: T_d, $d \in [D]$, has $Q_d(m', m)$ (a share of $Q(m', m)$, where Q is a ballot matrix of some voter) for all $1 \leq m' < m \leq M$.

forall $d \in [D]$ **do**
  forall $1 \leq m \leq M$ **do**
    T_d sets $Q_d(m, m) \leftarrow 0$
    forall $m < m' \leq M$ **do**
      T_d sets $Q_d(m', m) \leftarrow -Q_d(m, m')$
forall $1 \leq m' < m \leq M$ **do**
  Compute shares of $x := Q(m', m) \cdot (Q(m', m) + 1) \cdot (Q(m', m) - 1)$
  Recover x
  if $x \neq 0$ **then**
    Output "Illegal ballot" and **Abort**
forall $1 \leq m' < m \leq M$ **do**
  Compute shares of $\xi_{m',m} := 1_{Q(m',m)=0}$
  forall $1 \leq k \leq M$ **do**
    Compute shares of $\pi_{m',m,k} := \xi_{m',m} \cdot (Q(m', k) - Q(m, k))$
    Recover $\pi_{m',m,k}$
    if $\pi_{m',m,k} \neq 0$ **then**
      Output "Illegal ballot" and **Abort**
forall $1 \leq m \leq M$ **do**
  Compute shares of $\eta_m := \prod_{m'=1}^{m-1}(1 - \xi_{m',m})$
forall $1 \leq m \leq M$ **do**
  Compute shares of $Q_m := \sum_{m'=1}^{M} \eta_{m'} \cdot Q(m, m')$
forall $1 \leq m' < m \leq M$ **do**
  Compute shares of $\eta_m \eta_{m'}$
  Compute shares of $\gamma_{m',m} := (1 - \eta_m \eta_{m'}) + \eta_m \eta_{m'} \cdot (Q_{m'} - Q_m)$
  Compute shares of a nonzero random $r \in \mathbf{Z}_p$
  Compute shares of $x := r \cdot \gamma_{m',m}$
  Recover x
  if $x = 0$ **then**
    Output "The ballot is illegal" and **Abort**
Output "The ballot is legal" and **Terminate**

Output : The legality of the input ballot matrix Q.

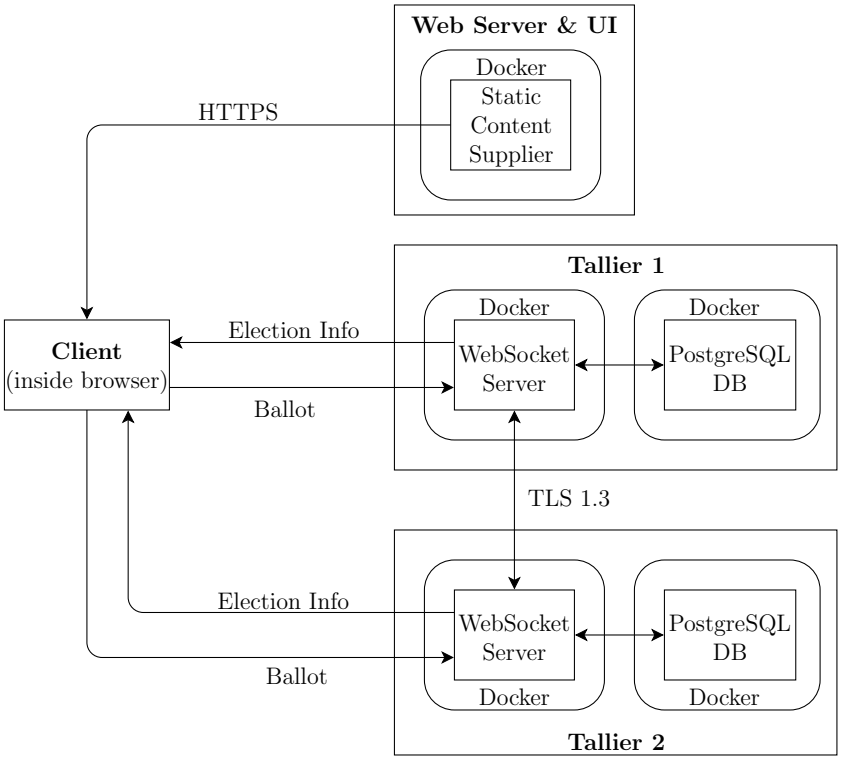

Figure 2: A high-level block diagram of the demo's system.

5.1. Web server and user interface

The Web server is implemented using the Flask and Jinja2 libraries, which render nearly static Web pages that serve as the user interface for the demo. Importantly, the Web server does not store any information or connect to the PostgreSQL database, thus ensuring a stateless and lightweight design. For real-world deployment, it is recommended to host the Web server on a separate machine in order to mitigate potential security risks.

The client-side code is developed using HTML and JavaScript. It is responsible for presenting to the voter the election form (through which the voter can conveniently input her or his preferred ranking). After the voter had submitted her vote, the client generates the corresponding ballot matrix, computes the corresponding secret shares according to the protocol, and distributes them to the talliers.

5.2. The tallier module

The tallier module is implemented using the websocket and asyncpg libraries. Communication between the user and any given tallier is done by an encrypted TLS 1.3 WebSocket channel, for ensuring secure data transmission. Users must register with all talliers, create an election, and submit their votes through this interface.

The MPC computations are implemented using async functions, which align with the natural structure of the protocol. This design closely mirrors implementations in referenced research papers, making the codebase easier to verify and extend.

To facilitate future development, a robust unit-testing setup has been implemented. This setup enables developers to verify the correctness of the implementation after modifications, and to streamline the development process.

5.3. User's usage flow

Upon entering the demo, new users are directed to a registration page, in which they fill out a registration form by providing their email address and full name. Consequently, a password will be sent to their email inbox, ensuring secure account setup. After registration, users return to the main page of the demo in which they can log in. The top bar on that page displays the communication status with each of the talliers, for real-time feedback on system connectivity.

Once logged in, users are directed to their dashboard, which displays all the elections they are managing or participating in as voters. (For new users, the list is empty.) From the dashboard, users can start a new election (for which they will be the managers), vote in an ongoing election (in which they are voters), and view the results of completed elections.

If a user wishes to start a new election, she must fill in an election creation form, see Figure 3. In that form she will configure various election parameters, including: election title, the voting rule (currently we implement two order-based rules, Copeland and Maximin), the names of all candidates, the desired number of winning candidates, and the emails of all eligible voters. The list of voters' emails can be provided either manually or by a CSV file in which each row holds an email address of a single voter.

Upon the successful creation of a new election, the election manager is redirected back to the dashboard. From there, the manager can start the

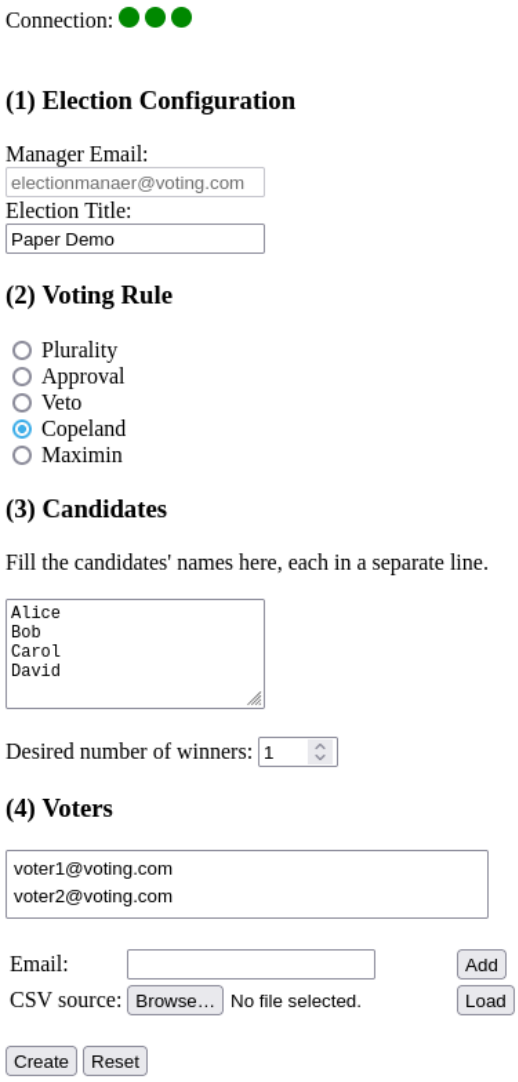

Figure 3: Election creation form

voting session for the election. By doing so, an email with election details and a link for submitting the vote will be sent to all eligible voters.

Each voter, upon accessing the voting link, fills in her preferences, by determining her ranking of the candidates (where ties are allowed), see Figure 4. The client translates that ranking to a corresponding ballot matrix and then generates secret shares of its (above diagonal) entries and distributes

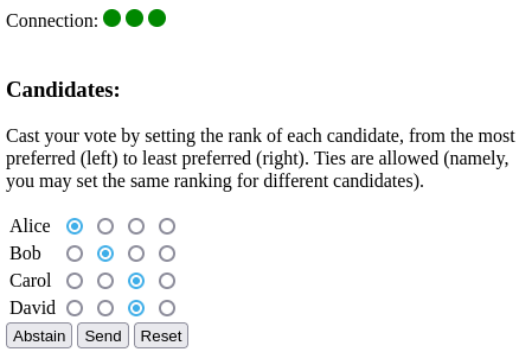

Figure 4: Voting page

them to the talliers. The talliers perform the proper validation checks, and if the ballot is verified they inform the voter that they had successfully received her ballot.

The election manager ends the election once the pre-determined election period had elapsed. At that point the talliers finalize the process by securely computing the winners and sending to all voters and to the manager an email with the election results.

5.4. Runtime evaluation

We have implemented our protocol over the field $\mathbf{Z}_p$ where $p = 2^{31} - 1$. As explained in Section 3.8, such a setting is sufficient for any conceivable election scenario.

The two parameters that affect the protocol's runtime are D, the number of talliers, and M, the number of candidates. Based on our discussion in Section 3.9 we set the value of D to $D \in \{3, 5, 7, 9\}$. The value of M was selected from the range $M \in \{3, 5, 10, 15, 20\}$. This range is representative of real-world elections, which rarely feature higher numbers of candidates or parties. In most national elections the number of competing entities remains modest—for example, parliamentary elections in many European countries typically involve fewer than 20 parties. Moreover, it is cognitively unrealistic to expect voters to meaningfully rank a large number of candidates. Empirical evidence suggests that individuals can comfortably evaluate and rank only a small number of alternatives (see, e.g., [33, 35]). Therefore, our choice of M values reflects both realistic election settings and voter capabilities.

As for the number of voters, N, it affects only the runtime for validating incoming ballots. The time for computing the final election results, on the

other hand, is independent of N. The runtimes reported in our experiments correspond to the cost of validating either a single ballot or a fixed-size batch of $b = 64$ ballots. Therefore, the total validation runtime for an election with N voters is obtained by multiplying the per-ballot cost by N, or multiplying the b-batch cost by N/b.

For each of the two voting rules, Copeland and Maximin, we measured the average runtime for validating the cast ballots, as well as measuring the time to compute the final election results at the end of the election period, and present their dependence on D and M. Indeed, the validation of a single ballot or a batch of ballots does not depend on N, while the computation of the final election results depends only on the dimension $M \times M$ of the aggregated ballot matrix $P = \sum_{n=1}^{N} P_n$, Eq. (1), but not on the number N of addends in the sum that defines it.

All experiments were executed on three servers that resided on the same LAN. Each of the servers had two AMD EPYC 7543 32-core processors, 128 GB RAM, and a "Gentoo Linux" operation system.

In the first experiment we measured the time for validating a single ballot in each of the two rules. As our implementation allows ties, the presented runtimes are for Sub-protocol 4. Recall that the same protocol is used for both rules − Copeland and Maximin. The results are shown in Figure 5. As can be seen, larger values of D entail computing more secret shares and sending more messages and, consequently, the overall runtime increases with D. As for M, the quadratic dependence of the runtime on M is clearly shown in the figure.

In the second experiment we measured the time for validating a batch of ballots, where we used batches of size 64. By validating several ballots in parallel, we can compute concurrently independent multiplications and thus reduce the overall runtime. In Figure 6 we show the ratio between the time to validate a batch of 64 ballots and the time to validate 64 single ballots. All shown values are smaller than 1 (namely, batching always saves time), where the more significant improvements (i.e., smaller ratios) were obtained for higher values of M and D. For example, if we concentrate on the highest parameters in our experiments, $M = 20$ and $D = 9$, then the average time to validate a single ballot, when using a 64-batch validation strategy, is roughly 50 msec, as implied by the values in Figures 5 and 6. That means that within a day it is possible to validate over 1,700,000 ballots. Clearly, those numbers can be further improved by using larger batches.

By choosing larger-size batches we could reduce the average runtime for

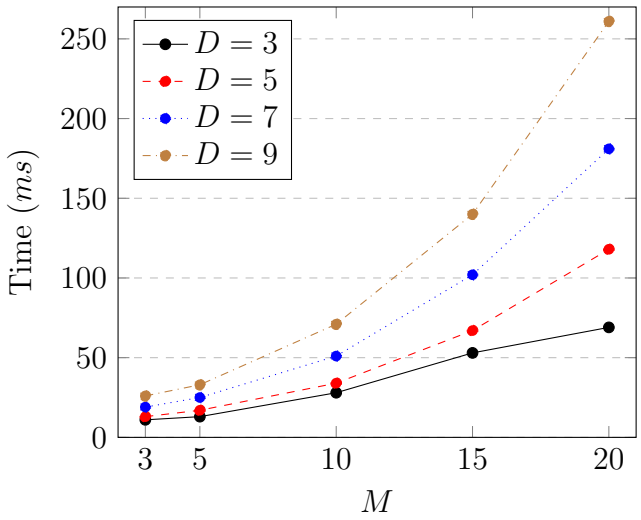

Figure 5: Runtimes (milliseconds) for validating a single ballot in each of the two rules, as a function of the number of candidates, M, and number of talliers, D.

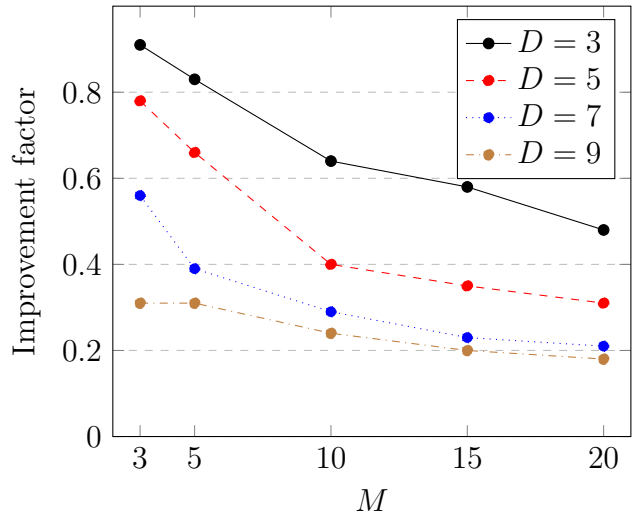

Figure 6: The improvement factor in runtime when validating batches of 64 ballots.

validating any single ballot even further. However, while selecting larger batch sizes translates to reduced computation times, it also translates to increased response times, since the validation of any single ballot will be delayed until a

sufficient number of ballots is received in order to form a batch of the desired size. If we wish to issue to the voter an immediate response regarding the validity of the ballot that he had submitted (so that in case the received ballot was not validated the voter would be asked to submit a new ballot), then the system should perform single ballot validation or set a short time window and then validate in parallel the batch of all ballots that were received during that window.

In the third and fourth experiments we measured the time to compute the winners in Copeland and Maximin rules; those runtimes are shown in Figures 7 and 8 respectively.

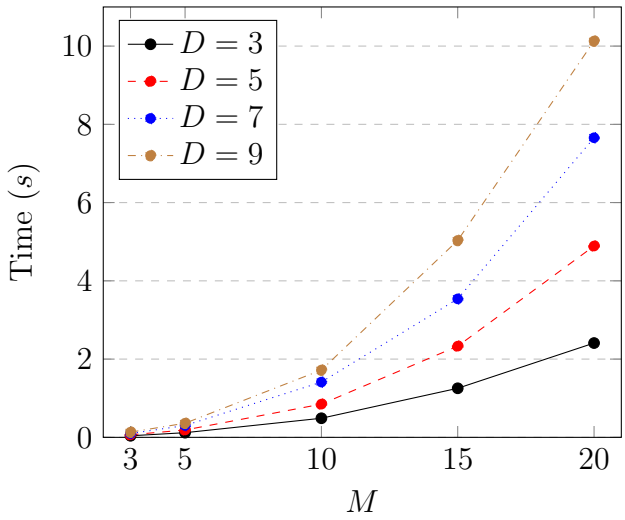

Figure 7: Runtimes (seconds) for computing election results in Copeland.

6. Related work

Secure e-voting can be approached using various cryptographic techniques. The earliest suggestion is that of Chaum [13], who suggested using a mix network (mixnet). The idea is to treat the ballots as ciphertexts. Voters encrypt their ballots and agents collect and shuffle these messages and thus anonymity of the ballots is preserved. Other studies followed and improved this model, e.g. Sako and Kilian [42], Adida [1], Boneh and Golle [6], Jakobsson et al. [30], Lee et al. [34], Neff [38]. However, while such systems preserve

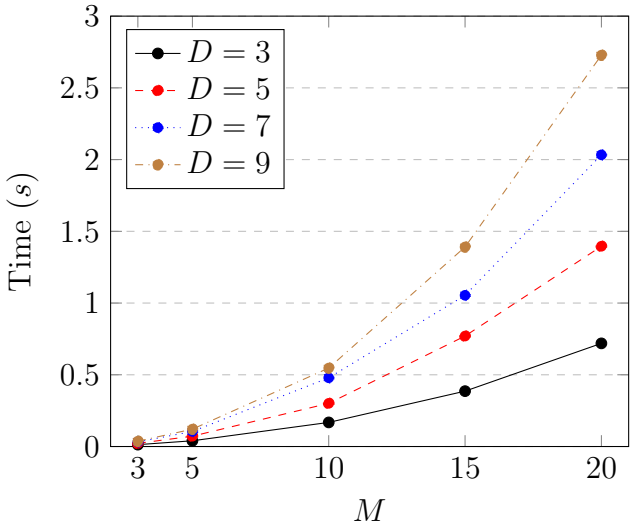

Figure 8: Runtimes (seconds) for computing election results in Maximin.

anonymity, the talliers are exposed to the actual ballots. The mere anonymity of the ballots might not provide sufficient security and this may encourage voters to abstain or vote untruthfully [21].

One of the approaches towards achieving tally-hiding privacy, and not just anonymity, is by employing homomorphic encryption, which allows performing computations on encrypted values without decrypting them first. The most common ciphers of that class are additively homomorphic, in the sense that the product of several ciphertexts is the encryption of the sum of the corresponding plaintexts. Such encryptions are suitable for secure voting, as was first suggested by Benaloh [3]. The main idea is to encrypt the ballots using a public-key homomorphic cipher. An agent aggregates the encrypted ballots and then sends an aggregated encrypted value to the tallier. The tallier decrypts the received ciphertexts and recovers the aggregation of the ballots, but is never exposed to the ballots themselves. Secure voting protocols that are based on homomorphic encryption were presented in e.g. [17, 18, 26, 29, 40, 41, 49].

While most studies on secure voting offered protocols for securing the voting process, some studies considered the question of private execution of the computation that the underlying voting rule dictates. We begin our survey with works that considered score-based voting rules. Canard et al.

[9] considered the MAJORITY JUDGMENT (MJ) voting rule [2]. They first translated the complex control flow and branching instructions that the MJ rule entails into a branchless algorithm; then they devised a privacy-preserving implementation of it using homomorphic encryption, distributed decryption schemes, distributed evaluation of Boolean gates, and distributed comparisons. Nair et al. [37] suggested to use secret sharing for the tallying process in PLURALITY voting [7]. Their protocol provides anonymity but does not provide perfect secrecy as it reveals the final aggregated score of each candidate. In addition, their protocol is vulnerable to cheating attacks, as it does not include means for detecting illegal votes. Küsters et al. [32] introduced a secure end-to-end verifiable tally-hiding e-voting system, called Ordinos, that implements the PLURALITY rule and outputs the K candidates that received the highest number of votes, or those with number votes that is greater than some threshold. Dery et al. [21] offered a solution based on MPC in order to securely determine the winners in elections governed by score-based voting rules, including PLURALITY, APPROVAL, VETO, RANGE and BORDA. Their protocols offer perfect privacy and very attractive runtimes.

Recently, few researchers have begun looking at order-based voting rules. Haines et al. [27] proposed a solution for the order-based SCHULZE rule [43]. Their solution does not preserve the privacy of voters who are indifferent between some pairs of candidates. In addition, their solution is not scalable to large election campaigns, as they report a runtime of 25 hours for an election with 10,000 voters. Hertel et al. [28] proposed solutions for COPELAND, MAXIMIN and SCHULZE voting rules. The evaluation of the SCHULZE method took 135 minutes for 5 candidates and 9 days, 10 hours, and 27 minutes for 20 candidates. Finally, Cortier et al. [16] considered the SINGLE TRANSFERABLE VOTE (STV) rule, which is a multi-stage rule, as well as SCHULZE rule. Even though their method is much more efficient than the one in Hertel et al. [28] for the SCHULZE rule, it is still not scalable to large election settings, as it took 8 hours and 50 minutes for $N = 1024$ voters and $M = 20$ candidates.

Our study is the first one that proposes protocols for order-based rules that are both fully private and lightweight so they offer a feasible and fast solution even for democracies of millions of voters, as our experiments indicate (see Section 5.4). The overwhelming advantage, in comparison to the above-mentioned recent works, is achieved mainly by our novel idea of distributed tallying.

7. Conclusion

Here we summarize our study (Section 7.1), describe a real-life implementation of the proposed protocol and system (Section 7.2), and outline future research directions (Section 7.3).

7.1. Summary

In this study we presented a protocol for the secure computation of order-based voting rules. Securing the voting process is an essential step toward a fully online voting process. A fundamental assumption in all secure voting systems that rely on fully trusted talliers (that is, talliers who receive the actual ballots from the voters) is that the talliers do not misuse the ballot information and that they keep it secret. In contrast, our protocol significantly reduces the trust vested in the talliers, as it denies the talliers access to the actual ballots and utilizes MPC techniques in order to compute the desired output without allowing any party access to the inputs (the private ballots). Even in scenarios where some (a minority) of the talliers betray that trust, privacy is ensured. Such a reduction of trust in the talliers is essential to increase the confidence of the voters in the voting system so that they would be further motivated to exercise their right to vote and, moreover, vote according to their true preferences, without fearing that their private vote will be disclosed to anyone.

Our protocol offers perfect ballot secrecy: the protocol outputs the identity of the winning candidates, but the voters as well as the talliers remain oblivious to any other related information, such as the actual ballots or any other value that is computed during the tallying process (e.g., how many voters preferred one candidate over the other). The design of a mechanism that offers perfect ballot secrecy must be tailored to the specific voting rule that governs the elections. We demonstrated our solution on the following order-based rules: Copeland, Maximin, Kemeny-Young and Modal-Ranking.[5] Ours is the first study that offers a fully private solution for order-based voting rules that is lightweight and practical for elections in real-life large democracies.

While several cryptographic voting protocols exist, they primarily focus on securing the casting and verification of ballots, rather than on computing complex voting outcomes under strict privacy constraints. To the best of our knowledge, there is currently no protocol that supports privacy-preserving

[5]The discussion of the latter two rules is deferred to the appendix.

computation of order-based voting rules with perfect ballot secrecy. Moreover, in related domains such as privacy-preserving collaborative filtering, prior art [45, 46] shows that secure multiparty computation based on secret sharing is significantly more efficient than comparable approaches using homomorphic encryption. This performance gap reinforces the practicality of our approach for real-world elections.

7.2. An implementation of the proposed protocol and system in the Gentoo council elections

The Gentoo Linux project (https://www.gentoo.org/) is a community-driven project with a decentralized governance model. Its two main organizational units are the Gentoo Foundation, which is a legal entity that holds trademarks, domains, and other assets related to Gentoo, and the Gentoo Council—a seven-member elected body that is responsible for policy decisions as well as resolving disputes.

The Gentoo Council holds an annual election for selecting seven council members from a pool of candidates. During a two-week voting period, the voters submit ranked ballots to three election officials (talliers). These officials then perform a transparent tallying process, *without ballot secrecy*, and publish the results on the organization's mailing list.

Through private communication with Gentoo officials, we learned that the original election rule was Schulze [43]. We demonstrated to the Gentoo officials our proposed voting model with a functional prototype implementation, which underwent a private evaluation. Following a successful vote by the election committee, they decided to adopt Copeland rule instead of Schulze's, using our system as a pilot program for their next election cycle.

7.3. Future research directions

The present study suggests several directions for future research:

(a) **Multi-winner elections.** Typical voting rules are usually oblivious to external social constraints, and they determine the identity of the winners solely based on private ballots. This is the case with the voting rules that we considered herein. In contrast, *multi-winner* election rules are designed specifically for selecting candidates that would satisfy the voters the most [23, 25], in the sense that they also comply with additional social conditions (e.g., that the selected winners include a minimal number of representatives of a specific gender, race, region, etc.). This problem has unique features, and therefore requires its own secure protocols. Examples of voting rules that

are designed for this purpose are Chamberlin-Courant [11] and Monroe [36].

(b) **A hierarchical tallier model.** We assumed a "flat" tallier model, where all talliers are operating on all ballots. However, in large voting systems, a hierarchical tallier may be more suitable. For example, in the US, it may be more suitable to use a hierarchy by county (first level), state (second level), and national (third and highest level). A modification of our protocol for such settings is in order.

(c) **Malicious talliers.** Our protocol assumes that the talliers are semi-honest, i.e., that they follow the prescribed protocol correctly. The semi-honesty of the talliers can be ensured in practice by securing the software and hardware of the talliers. However, it is possible to design an MPC protocol that would be immune even to malicious talliers that may deviate from the prescribed protocol. While such protocols are expected to have significantly higher runtimes, they could enhance even further the security of the system and the trust of voters in the preservation of their privacy.

Appendix A. Secure computations over secret shared data

In Section 2.2 we described the sub-protocols that our protocol utilizes. Here we provide an inside look at each of those four MPC sub-protocols.

Appendix A.1. Evaluating arithmetic functions

Let $f(x_1, \ldots, x_L)$ be an arithmetic function of L variables. Assume that the talliers hold D'-out-of-D shares of each of the inputs, $x_1, \ldots, x_L$. They wish to compute D'-out-of-D shares of $y := f(x_1, \ldots, x_L)$ without learning any information on any of the inputs nor on the output y. This challenge translates to the following two basic computational tasks: given D'-out-of-D shares of two secret values $u, v \in \mathbf{Z}_p$, compute D'-out-of-D shares of $au + bv$ (where a and b are public field elements), and of $u \cdot v$, without revealing u, v or the computed results ($au + bv$ and $u \cdot v$).

Let u_d and v_d denote the shares held by T_d, $d \in [D]$, corresponding to the two secrets u and v, respectively. Computing shares of the linear combination $au + bv$ is easy, since $\{au_d + bv_d : d \in [D]\}$ is a valid D'-out-of-D sharing of $au + bv$, as implied by the linearity of secret sharing. Consequently, each tallier can locally compute his share of the linear combination from his shares of the two inputs without any interaction with his peers.

Computing secret shares of the product $u \cdot v$ is more involved and requires the talliers to interact. In our protocol, we use the multiplication protocol proposed by Damgård and Nielsen [19], enhanced by a work by Chida et al. [14] that demonstrates some performance optimizations. We consider that computation as a black-box since the details of that computation are not relevant for our needs. Interested readers may refer to [19, 14] for more details.

Appendix A.2. Secure Comparison

Assume that $T_1, \ldots, T_D$ hold D'-out-of-D shares of two nonnegative integers u and v, where both u and v are smaller than p, which is the size of the underlying field $\mathbf{Z}_p$. The goal is to compute D'-out-of-D secret shares of the bit $1_{u<v}$ without learning any other information on u and v. Such a protocol is called secure comparison. In our protocol we used the secure comparison protocol proposed by Nishide and Ohta Nishide and Ohta [39], with some performance enhancements that we introduced.

Appendix A.3. Secure testing of positivity

Let u be an integer in the range $[-N, N]$. Assume that $T_1, \ldots, T_D$ hold D'-out-of-D shares of u, where the underlying field is $\mathbf{Z}_p$, and $p > 2N$. Our goal is to design an MPC protocol that allows the talliers to obtain D'-out-of-D shares of the bit $1_{u>0}$, without learning any additional information on u.

One way of solving such a problem would be to set $v = u + N$ and then test whether $N < v$ using the protocol of secure comparison from Section Appendix A.2. However, we propose a more efficient method for testing positivity. To this end, we state and prove the following lemma.

Lemma 5. *Under the above assumptions, $u > 0$ iff the LSB of $(-2u \mod p)$ is 1.*

Proof. Recall that $u \in [-N, N]$ and $N < \frac{p}{2}$. Assume that $u > 0$, namely, that $u \in (0, N]$. Hence, $-2u \in [-2N, 0)$. Therefore, as $2N < p$, $(-2u \mod p) = -2u + p$. As that number is odd, its LSB is 1. If, on the other hand, $u \leq 0$, then $u \in [-N, 0]$. Hence, $-2u \in [0, 2N] \subset [0, p-1]$. Therefore, $(-2u \mod p) = -2u$. As that number is even, its LSB is 0. □

Hence, the talliers can compute shares of $1_{u>0}$ by computing shares of the LSB of only one shared secret ($-2u$ in this case). In order to compute shares of $1_{u<v}$ using the protocol of [39] it is necessary to compute shares of three LSBs of shared secrets. Hence, computing shares of $1_{u>0}$ based on Lemma 5 is more efficient, roughly by a factor of 3, than computing such shares based on the general secure comparison protocol of [39]. In our protocol we compute shares of positivity bits based on this efficient computation.

Appendix A.4. Secure testing of equality to zero

Let u be an integer in the range $[-N, N]$. Assume that the talliers $T_1, \ldots, T_D$ hold D'-out-of-D shares of u, where the underlying field is $\mathbf{Z}_p$, and $p > 2N$. Our goal is to design an MPC protocol that enables the talliers to compute D'-out-of-D shares of the bit $1_{u=0}$, without learning any additional information on u.

A naïve approach would be to use the MPC positivity testing from Section Appendix A.3, once for u and once for $-u$. Clearly, $u = 0$ iff both of those tests fail. However, we can solve that problem in a more efficient manner, as we proceed to describe.

Fermat's little theorem states that if $u \in \mathbf{Z}_p \setminus \{0\}$ then $u^{p-1} = 1 \bmod p$. Hence, $1_{u\neq 0} = (u^{p-1} \bmod p)$. Therefore, shares of the bit $1_{u\neq 0}$ can be obtained by simply computing $u^{p-1} \bmod p$. The latter computation can be carried out by the square-and-multiply algorithm with up to $2\lceil \log p \rceil$ consecutive multiplications. Finally, as $1_{u=0} = 1 - 1_{u\neq 0}$, then shares of $1_{u\neq 0}$ can be readily translated into shares of $1_{u=0}$. The cost of the above described computation is significantly smaller than the cost of the alternative approach that performs positivity testing of both u and $-u$.

Appendix B. Other order-based rules

Here we consider two other order-based rules – Kemeny-Young [31, 51] and Modal Ranking [10], and describe secure protocols for implementing them.

Appendix B.1. Kemeny-Young

Appendix B.1.1. Description

The ballot matrices here, P_n, $n \in [N]$, are as in Maximin and, as before, $P = \sum_{n\in[N]} P_n$ is the aggregated ballot matrix. For every $1 \leq m \neq \ell \leq N$, $P(m, \ell)$ equals the number of voters who ranked C_m strictly higher than C_ℓ. The matrix P induces a score for each of the possible $M!$ rankings over $\mathbf{C} = \{C_1, \ldots, C_M\}$. Let $\rho = (\sigma_1, \ldots, \sigma_M)$, where $(\sigma_1, \ldots, \sigma_M)$ is a permutation of $\{1, \ldots, M\}$, be such a ranking. Here, for each $m \in [M]$, σ_m is the rank of C_m in the ranking. For example, if $M = 4$ then $\rho = (3, 1, 4, 2)$ is the ranking in which C_2 is the top candidate and C_3 is the least favored candidate. The score of a ranking ρ is defined as

$$w(\rho) = \sum_{\ell,m\in[M]:\sigma_m<\sigma_\ell} P(m, \ell)\,; \tag{B.1}$$

namely, one goes over all pairs of candidates C_m and C_ℓ such that ρ ranks C_m higher than C_ℓ (in the sense that $\sigma_m < \sigma_\ell$) and adds up the number of voters who agreed with this pairwise comparison. The ranking ρ with the highest score is selected, and the K leading candidates in ρ are the winners of the election.

Appendix B.1.2. Secure implementation

As in Protocol 1 for COPELAND and MAXIMIN, each voter V_n secret shares the entries of his ballot matrix P_n among the D talliers using a D'-out-of-D scheme. Since the ballot matrices here are as in MAXIMIN, where ties are allowed, their validation is carried out as described in Section 4.3.

After validating the cast ballots, the talliers add up their shares of P_n, for all validated ballot matrices, and then get D'-out-of-D shares of each of the non-diagonal entries in P.

To compute secret shares of the score of each possible ranking ρ, the talliers need only to perform summation according to Eq. (B.1). Note that that computation does not require the talliers to interact. Finally, it is needed to find the ranking with the highest score. That computation can be done by performing secure comparisons, as described in Section Appendix A.2.

Appendix B.2. Modal ranking

In MODAL RANKING, every voter submits a ranking of all candidates, where the ranking has no ties. Namely, if $R_{\mathbf{C}}$ is the set of all $M!$ rankings of the M candidates in $\mathbf{C}$, the ballot of each voter is a selection of one ranking from $R_{\mathbf{C}}$, as determined by his preferences. The rule then outputs the ranking that was selected by the largest number of voters (i.e., the mode of the distribution of ballots over $R_{\mathbf{C}}$). In case there are several rankings that were selected by the greatest number of voters, the rule outputs all of them, and then the winners are usually the candidates whose average position in those rankings is the highest.

The MODAL RANKING rule is an order-based voting rule over $\mathbf{C}$, but it is equivalent to the PLURALITY score-based voting rule (see Brandt et al. [7]) over the set of candidate rankings $R_{\mathbf{C}}$. Hence, it can be securely implemented by the protocol that was presented in Dery et al. [21].

References

[1] Ben Adida. 2008. Helios: Web-based Open-Audit Voting.. In *USENIX security symposium*, Vol. 17. 335–348.

[2] Michel Balinski and Rida Laraki. 2007. A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences* 104, 21 (2007), 8720–8725.

[3] Josh Benaloh. 1986. Secret sharing homomorphisms: Keeping shares of a secret secret. In *CRYPTO*. 251–260.

[4] Dan Bogdanov, Sven Laur, and Jan Willemson. 2008. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*. 192–206.

[5] G.R. Blakley. 1979. Safeguarding Cryptographic Keys. In *International Workshop on Managing Requirements Knowledge*. 313–317.

[6] Dan Boneh and Philippe Golle. 2002. Almost entirely correct mixing with applications to voting. In *CCS*. 68–77.

[7] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice*. Cambridge University Press.

[8] Felix Brandt and Tuomas Sandholm. 2005. Decentralized voting with unconditional privacy. In *AAMAS*. 357–364.

[9] Sébastien Canard, David Pointcheval, Quentin Santos, and Jacques Traoré. 2018. Practical strategy-resistant privacy-preserving elections. In *European Symposium on Research in Computer Security*. 331–349.

[10] Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. 2014. Modal ranking: A uniquely robust voting rule. In *AAAI*. 616-622.

[11] John R Chamberlin and Paul N Courant. 1983. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *The American Political Science Review* (1983), 718–733.

[12] David Chaum. 1988. Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA. In *EUROCRYPT*. 177–182.

[13] David L Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.

[14] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. 2018. Fast Large-Scale Honest-Majority MPC for Malicious Adversaries. In *CRYPTO*. 34–64.

[15] Arthur H Copeland. 1951. A reasonable social welfare function. In *Mimeographed notes from a Seminar on Applications of Mathematics to the Social Sciences, University of Michigan*.

[16] Véronique Cortier, Pierrick Gaudry, and Quentin Yang. 2022. A toolbox for verifiable tally-hiding e-voting systems. In *European Symposium on Research in Computer Security*. 631–652.

[17] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. 1997. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*. 103–118.

[18] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. 2010. A generalization of Pailliers public-key system with applications to electronic voting. *International Journal of Information Security* 9 (2010), 371–385.

[19] Ivan Damgård and Jesper Buus Nielsen. 2007. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*. 572–590.

[20] Herbert Aron David. 1963. *The method of paired comparisons*. Vol. 12. London.

[21] Lihi Dery, Tamir Tassa, and Avishay Yanai. 2021. Fear not, vote truthfully: Secure Multiparty Computation of score based rules. *Expert Systems with Applications* 168 (2021), 114434.

[22] Lihi Dery, Tamir Tassa, Avishay Yanai, and Arthur Zamarin. 2021. A Secure Voting System for Score Based Elections. In *CCS*. 2399–2401.

[23] Edith Elkind, Piotr Faliszewski, Jean-François Laslier, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2017. What Do Multiwinner Voting Rules Do? An Experiment Over the Two-Dimensional Euclidean Domain. In *AAAI*. 494-501.

[24] Piotr Faliszewski, Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. 2009. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research* 35 (2009), 275–341.

[25] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2017. Multiwinner voting: A new challenge for social choice theory. *Trends in computational social choice* 74 (2017), 27–47.

[26] Xingyue Fan, Ting Wu, Qiuhua Zheng, Yuanfang Chen, Muhammad Alam, and Xiaodong Xiao. 2020. HSE-Voting: A secure high-efficiency electronic voting scheme based on homomorphic signcryption. *Future Generation Computer Systems* 111 (2020), 754–762.

[27] Thomas Haines, Dirk Pattinson, and Mukesh Tiwari. Verifiable homomorphic tallying for the schulze vote counting scheme. In *Verified Software. Theories, Tools, and Experiments: 11th International Conference, VSTTE 2019, New York City, NY, USA, July 13–14, 2019, Revised Selected Papers 11*, pages 36–53. Springer, 2020.

[28] Fabian Hertel, Nicolas Huber, Jonas Kittelberger, Ralf Küsters, Julian Liedtke, and Daniel Rausch. 2021. Extending the Tally-Hiding Ordinos System: Implementations for Borda, Hare-Niemeyer, Condorcet, and Instant-Runoff Voting. In *E-Vote-ID 2021*. University of Tartu Press, 269–284.

[29] Alejandro Hevia and Marcos Kiwi. 2004. Electronic jury voting protocols. *Theoretical Computer Science* 321 (2004), 73–94.

[30] Markus Jakobsson, Ari Juels, and Ronald L Rivest. 2002. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX*. 339–353.

[31] John G Kemeny. 1959. Mathematics without numbers. *Daedalus* 88, 4 (1959), 577–591.

[32] Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. 2020. Ordinos: A verifiable tally-hiding e-voting system. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. 216–235.

[33] Richard R Lau and David P Redlawsk. Advantages and disadvantages of cognitive heuristics in political decision making. *American journal of political science*, pages 951–971, 2001.

[34] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. 2003. Providing receipt-freeness in mixnet-based voting protocols. In *International conference on information security and cryptology*. 245–258.

[35] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[36] Burt L Monroe. 1995. Fully proportional representation. *American Political Science Review* (1995), 925–940.

[37] Divya G. Nair, V. P. Binu, and G. Santhosh Kumar. 2015. An Improved E-voting scheme using Secret Sharing based Secure Multi-party Computation. *CoRR* abs/1502.07469 (2015).

[38] C Andrew Neff. 2001. A verifiable secret shuffle and its application to e-voting. In *CCS*. 116–125.

[39] Takashi Nishide and Kazuo Ohta. 2007. Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In *PKC*. 343–360.

[40] J Chandra Priya, Ponsy RK Sathia Bhama, S Swarnalaxmi, A Aisathul Safa, and I Elakkiya. 2018. Blockchain centered homomorphic encryption: A secure solution for E-balloting. In *International conference on Computer Networks, Big data and IoT*. 811–819.

[41] Fatemeh Rezaeibagha, Yi Mu, Shiwei Zhang, and Xiaofen Wang. 2019. Provably secure (broadcast) homomorphic signcryption. *International Journal of Foundations of Computer Science* 30, (2019), 511–529.

[42] Kazue Sako and Joe Kilian. 1995. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*. 393–403.

[43] Markus Schulze. 2011. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social choice and Welfare* 36, (2011), 267–303.

[44] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22 (1979), 612–613.

[45] Erez Shmueli and Tamir Tassa. Mediated secure multi-party protocols for collaborative filtering. *ACM Trans. Intell. Syst. Technol.*, 11:15:1–15:25, 2020.

[46] Tamir Tassa and Alon Ben Horin. Privacy-preserving collaborative filtering by distributed mediation. *ACM Trans. Intell. Syst. Technol.*, 13:102:1–102:26, 2022.

[47] Paul B Simpson. 1969. On defining areas of voter choice: Professor Tullock on stable voting. *The Quarterly Journal of Economics* (1969), 478–490.

[48] Alan Szepieniec and Bart Preneel. 2015. New techniques for electronic voting. In *IACR Cryptol. ePrint Arch.*. 809.

[49] Xuechao Yang, Xun Yi, Surya Nepal, Andrei Kelarev, and Fengling Han. 2018. A secure verifiable ranked choice online voting system based on homomorphic encryption. *IEEE Access* 6 (2018), 20506–20519.

[50] A.C. Yao. 1982. Protocols for secure computation. In *FOCS*. 160–164.

[51] Peyton Young. 1995. Optimal voting rules. *Journal of Economic Perspectives* 9, (1995), 51–64.