

# Towards Secure Virtual Elections: Multiparty Computation of Order Based Voting Rules

Tamir Tassa <sup>\*</sup>      Lihi Dery <sup>†</sup>

May 9, 2024

## 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 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. Evaluation of the protocol's computational costs establishes that it is lightweight and can be readily implemented in real-life electronic elections. **Keywords:** Secure voting, Perfect ballot secrecy, Multiparty computation, Computational social choice, Virtual elections

## 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. One of the

---

<sup>\*</sup>The Open University, Israel, tamirta@openu.ac.il

<sup>†</sup>Ariel University, Israel, lihid@ariel.ac.il

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 be deemed benign or even desired, in some cases, it may be problematic. For example, in small-scale elections (say, elections to university senates), the full tally may put the voters' privacy at risk; say, if Alice promised Bob to vote for him, but at the end of the day, no one had voted for him, Alice's failure to keep her promise to Bob would be exposed. Such potential privacy breaches may be an obstacle for some voters to vote truthfully. In some cases, exposing the full tally might severely damage the image of some candidates who may, consequently, refrain from submitting their candidacy again in future elections. Such problems can be averted if the voting system is tally-hiding (Szepieniec and Preneel, 2015); namely, if it reveals only the desired final results and nothing else. 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 design a tally-hiding voting system that offers perfect ballot secrecy, or full privacy (Chaum, 1988); i.e., 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 full privacy may reduce the possibility of voters' coercion and also increase the voters' confidence that their vote remains secret. Hence, full privacy 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 (Brandt and Sandholm, 2005).

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 that end is *homomorphic encryption*, namely, a form of encryption that preserves 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. 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 amongst 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.

**Our contributions.** We consider elections that are governed by an order-based voting rule. We focus on two such rules – COPELAND (Copeland, 1951), and MAXIMIN (Simpson, 1969) (a.k.a KRAMER-SIMPSON). 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 Appendices G and H we describe two other order-based rules, KEMENY-YOUNG (Kemeny, 1959; Young, 1995) and MODAL RANKING (Caragiannis et al., 2014), 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 foundations on which our protocol is based. In Section 3 we present

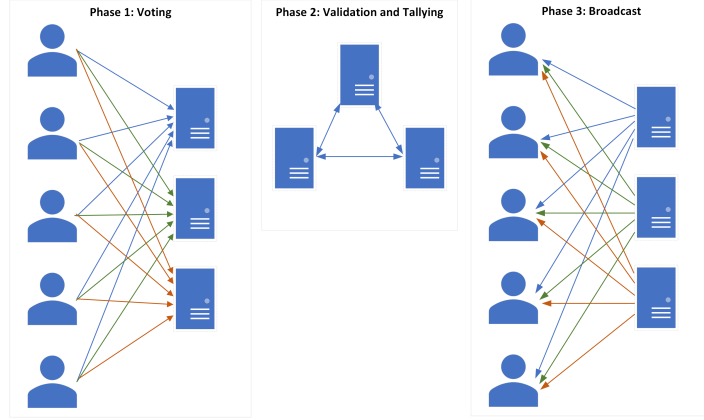


Figure 1: A high-level description of the protocol.

our secure voting protocol for COPELAND and MAXIMIN rules and then, in Section 4, we evaluate its computational costs. A survey of related work is given in Section 5, and we conclude in Section 6.

## 2 Cryptographic preliminaries

Our protocol relies heavily on cryptographic machinery. This section provides the required cryptographic background and an explanation of how the presented techniques are implemented in our protocol. In Section 2.1 we provide a brief introduction to secret sharing. In Section 2.2 we describe the cryptographic principles of our protocol. Then, in the following sections (Sections 2.3-2.6) we provide the details of specific computations that we use in our protocol.

We note that while Sections 2.1 and 2.2 are necessary for understanding our protocol that is presented in Section 3, they are also sufficient, in the sense that readers who are less interested in the cryptographic details may skip Sections 2.3-2.6 and still be able to fully understand the secure voting protocol.

## 2.1 Secret sharing

Secret sharing schemes (Shamir, 1979) enable distributing a secret  $s$  among a set of parties,  $\mathbf{T} = \{T_1, \dots, T_D\}$ . Each party,  $T_d$ ,  $d \in [D] := \{1, \dots, D\}$ , is given a random value  $s_d$ , called *a share*, that relates to the secret  $s$ . Those shares satisfy the following conditions: (a)  $s$  can be reconstructed only by combining the shares given to specific subsets of parties, which are called *authorized subsets* (the collection of all authorized subsets is called *the access structure*); (b) shares held by any other subset of parties reveal zero information on  $s$ .

The notion of secret sharing was introduced, independently, by Shamir (1979) and Blakley (1979), for the case of threshold secret sharing. In threshold secret sharing there is some threshold  $D' \leq D$  and then the access structure consists of all subsets 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  $\mathbb{Z}_p$ , where  $p > D$  is a prime sufficiently large so that all possible secrets may be represented in  $\mathbb{Z}_p$ . It has two procedures: **Share** and **Reconstruct**:

- **Share** $_{D',D}(x)$ . The procedure samples a uniformly random polynomial  $g(\cdot)$  over  $\mathbb{Z}_p$ , of degree at most  $D' - 1$ , where the free coefficient is the secret  $s$ . That is,  $g(x) = s + a_1x + a_2x^2 + \dots + a_{D'-1}x^{D'-1}$ , where  $a_j$ ,  $1 \leq j \leq D' - 1$ , are selected independently and uniformly at random from  $\mathbb{Z}_p$ . The procedure outputs  $s_d = g(d)$ ,  $d \in [D]$ , where  $s_d$  is the share given to the  $d$ th party,  $T_d$ ,  $d \in [D]$ .

- **Reconstruct** $_{D'}(s_1, \dots, s_D)$ . The procedure is given any selection of  $D'$  shares out of  $\{s_1, \dots, s_D\}$ ; it then interpolates a polynomial  $g(\cdot)$  of degree at most  $D' - 1$  using the given points and 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.

We note that there are secret sharing schemes that realize more general access structures, e.g, Brickell (1989); Farràs et al. (2012); Shamir (1979); Tassa and Dyn (2006), but for our purposes the basic Shamir's scheme suffices.

## 2.2 The cryptographic principles of our protocol

Our protocol involves two sets of parties: *voters*,  $\mathbf{V} = \{V_1, \dots, V_N\}$ , and *talliers*,  $\{T_1, \dots, T_D\}$ . The talliers are assumed to be semi-honest, i.e., they follow the prescribed protocol, but try to extract from their view in the protocol information on the private inputs of the voters. We also assume them to have an *honest majority*, in the sense that if some of them are corrupted by a malicious adversary, their number is smaller than

$$D' := \lfloor (D + 1)/2 \rfloor. \quad (1)$$

On the other hand, the voters (any number of them) may be malicious. Hence, our protocol includes mechanisms to detect potential deviations from the prescribed protocol in order to ensure that the computed election results correctly identify the winners as dictated by all legal ballot and the underlying rule. During the entire process, the sensitive information that should be kept secret are the private ballots as well as all intermediate computed values; nothing beyond the identity of the winner(s) should be leaked.

In the protocol, each voter creates shares of his<sup>1</sup> private ballot and distributes them to the  $D$  talliers. As those ballots are matrices (see the definitions in Section 3), the secret sharing is carried out for each matrix entry independently.

We use Shamir's  $D'$ -out-of- $D$  secret sharing, where hereinafter  $D'$  is as defined in Eq. (1). With that selection of the threshold  $D'$ , at least half of the talliers would need to collude in order to recover the secret ballots. Under our assumption that the set of talliers has an honest majority, our protocol provides information-theoretic security. Higher values of  $D$  will imply greater security against coalitions of corrupted talliers, but they will also imply higher costs.

The first thing that the talliers need to do when receiving from a voter his ballot matrix shares 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. Those 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

---

<sup>1</sup>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.

perform those computations on the distributed shares, without revealing the shares to each other. As we shall see later on, those computations boil down to the four specific tasks that we proceed to describe.

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

1. If  $y = f(x_1, \dots, x_L)$ , where  $f$  is some public arithmetic function, compute  $D'$ -out-of- $D$  shares in  $y$ .
2. If  $y = 1_{x_1 < x_2}$  is the bit that equals 1 if  $x_1 < x_2$  and 0 otherwise<sup>2</sup>, compute  $D'$ -out-of- $D$  shares in  $y$ .
3. If  $p > 2N$  ( $N$  is the number of voters) and  $x_1 \in [-N, N]$ , compute  $D'$ -out-of- $D$  shares in the bit  $1_{x_1 > 0}$ .
4. If  $p > 2N$  and  $x_1 \in [-N, N]$ , compute  $D'$ -out-of- $D$  shares in the bit  $1_{x_1 = 0}$ .

In the next subsections, we describe sub-protocols of secure multiparty computation (MPC) (Yao, 1982) for performing those computations.

## 2.3 Evaluating polynomials

Let  $f(x_1, \dots, x_L)$  be an  $L$ -variate polynomial. Assume that the talliers have  $D'$ -out-of- $D$  shares in each of the  $L$  inputs and they wish to compute  $D'$ -out-of- $D$  shares in the output  $y = f(x_1, \dots, x_L)$ . To do so, it is necessary to solve the following problems: given  $D'$ -out-of- $D$  shares in two secret values  $u, v \in \mathbb{Z}_p$ , compute  $D'$ -out-of- $D$  shares in  $au + bv$ , where  $a, b$  are public field elements, and in  $u \cdot v$ , without reconstructing any of those values ( $u, v, au + bv$  and  $u \cdot v$ ).

Let  $u_d$  and  $v_d$  denote the shares held by  $T_d$ ,  $d \in [D]$ , in the two input values  $u$  and  $v$ , respectively. The linearity of secret sharing implies that  $au_d + bv_d$ ,  $d \in [D]$ , are proper  $D'$ -out-of- $D$  shares in  $au + bv$ . Hence, linear combinations may be resolved without any interaction amongst the talliers.

---

<sup>2</sup>Hereinafter, for any predicate  $\Pi$ , we let  $1_\Pi$  denote the binary variable that equals 1 iff (if and only if) the predicate  $\Pi$  holds.

The procedure for resolving multiplication is trickier and requires the talliers to interact. In our protocol, we use the construction proposed by Damgård and Nielsen (2007), enhanced by a work by Chida et al. (2018) that demonstrates some performance optimizations. The details of this computation are not essential for understanding our secure voting protocol, but for the sake of completeness we describe the computation in Appendix A.

The complexity of securely evaluating arithmetic functions is measured by the number of multiplications (as those are the operations that require significantly more computational and communication costs), and the degree of the polynomial (as it determines the number of rounds of communication).

## 2.4 Secure Comparison

Assume that  $T_1, \dots, T_D$  hold  $D'$ -out-of- $D$  shares in two integers  $a$  and  $b$ , where both  $a$  and  $b$  are smaller than  $p$ , which is the size of the underlying field  $\mathbb{Z}_p$ . They wish to compute  $D'$ -out-of- $D$  secret shares in the bit  $1_{a < b}$  without learning any other information on  $a$  and  $b$ . A protocol that does that is called *secure comparison*.

Nishide and Ohta (2007) presented a method for secure comparison that is based on the following simple observation. Let us denote the bits  $1_{a < \frac{p}{2}}$ ,  $1_{b < \frac{p}{2}}$ ,  $1_{[(a-b) \bmod p] < \frac{p}{2}}$ , and  $1_{a < b}$  by  $w$ ,  $x$ ,  $y$ ,  $z$ , respectively. Then

$$\begin{aligned} z &= w(1 - x) + (1 - w)(1 - x)(1 - y) + wx(1 - y) \\ &= 1 - x - y + xy + w(x + y - 2xy). \end{aligned} \tag{2}$$

Hence, we reduced the problem of comparing two secret shared values,  $a$  and  $b$ , to computing three other comparison bits —  $w, x, y$ , and then evaluating an arithmetic function of them, Eq. (2). What makes this expression efficiently computable is the fact that in the comparison bits,  $w = 1_{a < \frac{p}{2}}$ ,  $x = 1_{b < \frac{p}{2}}$ , and  $y = 1_{[(a-b) \bmod p] < \frac{p}{2}}$ , the right-hand side is  $\frac{p}{2}$ , as we proceed to explain.

**Lemma 1.** *Given a finite field  $\mathbb{Z}_p$  and a field element  $q \in \mathbb{Z}_p$ , then  $q < \frac{p}{2}$  iff the least significant bit (LSB) of  $(2q \bmod p)$  is zero.*

(The proofs of all lemmas and theorems are given in Appendices B–F.)

In view of Lemma 1, the talliers may compute  $D'$ -out-of- $D$  shares in  $w = 1_{a < \frac{p}{2}}$  (and similarly for  $x = 1_{b < \frac{p}{2}}$ , and  $y = 1_{[(a-b) \bmod p] < \frac{p}{2}}$ ) as follows: each  $T_d$  will translate the share he holds in  $a$  to a share in  $2a$  by multiplying



the share by 2; then, all talliers will use their shares in  $2a$  in order to compute shares in the LSB of  $2a$  Nishide and Ohta (2007).

We conclude this section by commenting on the complexity of the above described secure comparison protocol. Computing shares in the LSB of a shared value requires 13 rounds of communication and  $93\ell + 1$  multiplications, where hereinafter  $\ell = \log_2(p)$ . Since we have to compute three such bits (i.e.,  $w$ ,  $x$ , and  $y$ ) then we can compute shares of those three bits in 13 rounds and a total of  $279\ell + 3$  multiplications. Finally, we should evaluate the expression in Eq. (2), which entails two additional rounds and two additional multiplications. Hence, the total complexity is 15 rounds and  $279\ell + 5$  multiplications.

## 2.5 Secure testing of positivity

Let  $x$  be an integer in the range  $[-N, N]$ . Assume that  $T_1, \dots, T_D$  hold  $D'$ -out-of- $D$  shares in  $x$ , where the underlying field is  $\mathbb{Z}_p$ , and  $p > 2N$ . Our goal here is to design an MPC protocol that will issue to  $T_1, \dots, T_D$   $D'$ -out-of- $D$  shares in the bit  $1_{x>0}$ , without learning any further information on  $x$ .

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

Lemma 2 implies that testing the positivity of a secret requires a single LSB computation. Hence, in view of the discussion in Section 2.4, the computational cost of that task is 13 rounds of communication and  $93\ell + 1$  multiplications.

## 2.6 Secure testing of equality to zero

Let  $x$  be an integer in the range  $[-N, N]$ . Assume that  $T_1, \dots, T_D$  hold  $D'$ -out-of- $D$  shares in  $x$ , where the underlying field is  $\mathbb{Z}_p$ , and  $p > 2N$ . Our goal here is to design an MPC protocol that will issue to  $T_1, \dots, T_D$   $D'$ -out-of- $D$  shares in the bit  $1_{x=0}$ , without learning any further information on  $x$ . It is possible to solve that problem by implementing the MPC positivity testing that we described above in Section 2.5, once for  $x$  and once for  $-x$ . Clearly,  $x = 0$  iff both of those tests fail. However, we can solve that problem in a much more efficient manner, as we proceed to describe.

Fermat's little theorem states that if  $x \in \mathbb{Z}_p \setminus \{0\}$  then  $x^{p-1} = 1 \bmod p$ . Hence,  $1_{x \neq 0} = (x^{p-1} \bmod p)$ . Therefore, shares in the bit  $1_{x \neq 0}$  can

be obtained by computing  $x^{p-1} \bmod p$ . The latter computation can be carried out by the square-and-multiply algorithm with up to  $2\ell$  consecutive multiplications, where, as before,  $\ell = \log p$ . Finally, as  $1_{x=0} = 1 - 1_{x \neq 0}$ , then shares in  $1_{x \neq 0}$  can be readily translated into shares in  $1_{x=0}$ . The cost of the above described computation is significantly smaller than the cost of the alternative approach that performs positivity testing of both  $x$  and  $-x$ .

### 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. 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, without recovering the actual ballot. The characterization that we describe in Section 3.2 will be used later on to perform such an oblivious validation.

Then, in Section 3.3 we introduce our secure voting protocol. That protocol description includes a sub-protocol for validating the cast ballots and sub-protocols 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 of the overall security of our protocol (Section 3.8) and its implementation in an end-to-end voting system (Section 3.9).

#### 3.1 Formal definitions

We consider a setting in which there are  $N$  voters,  $\mathbf{V} = \{V_1, \dots, V_N\}$ , that wish to hold an election over  $M$  candidates,  $\mathbf{C} = \{C_1, \dots, C_M\}$ , and select  $1 \leq K < M$  out of them. 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 *ballot 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 *aggregated ballot matrix*,

$$P = \sum_{n=1}^N P_n, \quad (3)$$

induces the following score for each candidate:

$$\mathbf{w}(m) := |\{m' \neq m : P(m, m') > 0\}| + \alpha |\{m' \neq m : P(m, m') = 0\}|. \quad (4)$$

Namely,  $\mathbf{w}(m)$  equals the number of candidates  $C_{m'}$  that a majority of the voters ranked lower than  $C_m$ , plus  $\alpha$  times the number of candidates  $C_{m'}$  who broke even with  $C_m$ . The parameter  $\alpha$  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  $\alpha$  is known as COPELAND $^{\frac{1}{2}}$  (Faliszewski et al., 2009).

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. (3). 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  $\mathbf{w}(m) := \min_{m' \neq m} P(m, m')$ .

### 3.2 Characterization of legal ballot matrices in the COPELAND and MAXIMIN rules

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 3.** *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. The set  $\{Q_m := \sum_{m' \in [M]} Q(m', m)\}_{m \in [M]}$  consists of  $M$  distinct values.

**Theorem 4.** *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. The set  $\{Q_m := \sum_{m' \in [M]} Q(m', m)\}_{m \in [M]}$  consists of  $M$  distinct values.

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

$$\Gamma(Q) := (Q(m, m') : 1 \leq m < m' \leq M).$$

Conditions 2 and 3 in Theorems 3 and 4 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 A secure voting 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.

Phase 1 (Lines 1-8) is the voting phase. Here, each voter  $V_n$ ,  $n \in [N] := \{1, \dots, N\}$ , constructs his ballot matrix,  $P_n$  (Line 2). He then samples a random share-generating polynomial of degree  $D' - 1$  for each of the  $M(M-1)/2$  entries in  $\Gamma(P_n)$ , where  $\Gamma$  is the projection mapping defined in Section 3.2 (Lines 3-5). Finally, he sends to each tallier his relevant share in each of those entries, namely, the value of the corresponding share-generating polynomial at  $x = d$ ,  $d \in [D]$  (Lines 6-7). Following that, the talliers engage in an MPC sub-protocol to verify the legality of  $V_n$ 's ballot, without actually recovering that ballot (Line 8). We describe the validation sub-protocol in Section 3.4. Ballots that are found to be illegal are discarded.

Phase 2 of the protocol (Lines 9-11) is carried out after the voting phase had ended. 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. (3), for all  $1 \leq m < m' \leq M$ . The heart of the protocol is in Line 11: here, the talliers engage in an MPC sub-protocol in order to find the indices of the  $K$  winning candidates. We describe the details of those computations in Sections 3.6 and 3.7.

---

**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  $\mathbf{V}$ .

```
1 forall  $V_n, n \in [N]$ , do
2   Construct the ballot matrix,  $P_n$ , according to the selected
   indexing of candidates and the voting rule;
3   forall  $1 \leq m < m' \leq M$  do
4     Select uniformly at random  $a_{n,m,m',j} \in \mathbb{Z}_p, 1 \leq j \leq D' - 1$ ;
5     Set  $g_{n,m,m'}(x) = P_n(m, m') + \sum_{j=1}^{D'-1} a_{n,m,m',j} x^j$ ;
6   forall  $d \in [D]$  do
7     Send to  $T_d$  the set  $\{n, m, m', g_{n,m,m'}(d) : 1 \leq m < m' \leq M\}$ ;
8   After all talliers receive their shares in  $V_n$ 's ballot, they engage in
   an MPC sub-protocol to check its legality;
9 forall  $T_d, d \in [D]$  do
10  Set  $G_d(m, m') = \sum_{n \in [N]} g_{n,m,m'}(d)$ , for all  $1 \leq m < m' \leq M$ ;
11  $T_1, \dots, T_D$  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, or in order to sabotage the proper aggregation of the ballots. 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 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 sub-protocol 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, it is possible to notify the voter that his ballot was rejected and allow him to resubmit it once again. If the resubmitted ballot is still 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 actions against dishonest voters, might deter voters from attempting to cheat 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 3 and 4 for COPELAND and MAXIMIN, respectively. Note that the talliers need only to verify conditions 1 and 4; condition 2 needs no verification since the voters do not distribute shares in 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.

**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 in  $x$  to compute the product  $(x+1) \cdot (x-1)$  for COPELAND or  $x \cdot (x-1)$  for MAXIMIN, as described in Section 2.3. If the computed product is zero for each of the  $M(M-1)/2$  entries of  $\Gamma(Q)$ , then  $\Gamma(Q)$  satisfies condition 1 in Theorem 3 (COPELAND) or in Theorem 4 (MAXIMIN). If, on the other hand, some of the multiplication gates issue a nonzero output, then the ballot will be rejected.

**Verifying condition 4.** First, we make the following observation. Let  $x, y \in \mathbb{Z}_p$  be two values that are shared among the talliers. Denote by  $x_d$  and  $y_d$  the  $D'$ -out-of- $D$  shares that  $T_d$  has in  $x$  and  $y$ , respectively. Then if  $a, b \in \mathbb{Z}_p$  are two publicly known field elements, it is easy to see that  $ax_d + by_d$  is a proper  $D'$ -out-of- $D$  share in  $ax + by$ ,  $d \in [D]$ . Also  $a + x_d$  is a proper  $D'$ -out-of- $D$  share in  $a + x$ .

Using the above observation regarding the linearity of secret sharing, then once the talliers receive  $D'$ -out-of- $D$  shares in each entry in  $\Gamma(Q)$ , they can proceed to compute  $D'$ -out-of- $D$  shares in the corresponding column sums,  $Q_m$ ,  $m \in [M]$ , as we proceed to show.

In COPELAND, conditions 2 and 3 in Theorem 3 imply that

$$Q_m = \sum_{m' \in [M]} Q(m', m) = \sum_{m' < m} Q(m', m) - \sum_{m < m'} Q(m, m'). \quad (5)$$

Since the talliers hold shares in  $Q(m', m)$  for all  $1 \leq m' < m \leq M$ , they can use Eq. (5) and the linearity of secret sharing to compute shares in  $Q_m$ ,  $m \in [M]$ . In MAXIMIN, on the other hand, conditions 2 and 3 in Theorem 4

imply that

$$Q_m = \sum_{m' < m} Q(m', m) - \sum_{m < m'} Q(m, m') + (M - m). \quad (6)$$

Here too, the linearity of sharing and the relation in Eq. (6) enable the talliers to compute shares in  $Q_m$ ,  $m \in [M]$ , 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

$$F(Q) := \left( \prod_{1 \leq m' < m \leq M} (Q_m - Q_{m'}) \right)^2. \quad (7)$$

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

**Privacy of the verification sub-protocol.** 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 3 and 4 offers perfect privacy for honest voters. If the ballot  $Q$  is legal then all multiplication gates will issue a zero output. 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 3 and 4 also offers perfect privacy, as we proceed to argue. If  $Q$  is a valid ballot in COPELAND, then the ordered tuple  $(Q_1, \dots, Q_M)$  is a permutation of the ordered tuple  $(-M+1, -M+3, \dots, M-3, M-1)$ . This statement follows from the proof of condition 4 in Theorem 3. Hence, as can be readily verified,

$$\prod_{1 \leq m' < m \leq M} (Q_m - Q_{m'}) = \pm 2^{\binom{M}{2}} \cdot \prod_{1 \leq m' < m \leq M} (m - m'), \quad (8)$$

where the sign of the product in Eq. (8) is determined by the signature of  $(Q_1, \dots, Q_M)$  when viewed as a permutation of the ordered tuple  $(-M+1, -M+3, \dots, M-3, M-1)$ . Hence, as the talliers compute  $F(Q)$ , Eq. (7),

which equals the square of the product on the left hand side of Eq. (8), then for *any* legal ballot they will always recover the same value, which is

$$F(Q) = \left( 2^{\binom{M}{2}} \cdot \prod_{1 \leq m' < m \leq M} (m - m') \right)^2.$$

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

$$F(Q) = \left( \prod_{1 \leq m' < m \leq M} (m - m') \right)^2.$$

**Computational cost.** Verifying condition 1 can be performed in parallel for all  $M(M - 1)/2$  entries in a given ballot, and also for several different ballots. Hence, in order to perform a batch validation of  $B$  ballots, the talliers need to compute  $BM(M - 1)/2$  simultaneous multiplication gates.

The verification of condition 4 over a single ballot requires performing a sequence of  $M(M - 1)/2$  multiplications. Hence, in order to perform a batch validation of  $B$  ballots, the talliers need to go through  $M(M - 1)/2$  rounds, where in each round they compute  $B$  simultaneous multiplication gates.

### 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  $\{s_1, \dots, s_D\}$  are the shares that  $V$  distributed to  $T_1, \dots, T_D$  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). To that end the talliers proceed as follows:



1. Each tallier  $T_d$ ,  $d \in [D]$ , produces a random number  $r_d$  and distributes to all talliers  $D'$ -out-of- $D$  shares in it, denoted  $\{r_{d,1}, \dots, r_{d,D}\}$ .
2. Each  $T_d$ ,  $d \in [D]$ , computes  $\hat{s}_d = s_d + \sum_{j \in [D]} r_{j,d}$ . Namely,  $T_d$  adds to the share that he had received from  $V$  in the secret entry in  $V$ 's ballot matrix the shares received from all talliers in the random numbers that they had produced.
3. Each  $T_d$ ,  $d \in [D]$ , broadcasts  $\hat{s}_d$ .
4. All talliers compute a polynomial  $f$  of degree up to  $D - 1$  such that  $f(d) = \hat{s}_d$ ,  $d \in [D]$ .
5. If  $\deg f \leq D' - 1$  then the set of shares  $\{s_1, \dots, s_D\}$  that  $V$  had distributed is a legal  $D'$ -out-of- $D$  sharing of some scalar in the field.
6. Otherwise, the ballot is rejected.

**Lemma 5.** *The above procedure is correct and fully preserves the voter's privacy.*

We note that the first step 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 in them to be used later on in masking the  $M(M - 1)/2$  entries in each voter's ballot matrix.

### 3.6 Computing the winners in COPELAND rule

The parameter  $\alpha$  in Eq. (4) is always a rational number; typical settings of  $\alpha$  are 0, 1, or  $\frac{1}{2}$  (Faliszewski et al., 2009). Assume that  $\alpha = \frac{s}{t}$  for some integers  $s$  and  $t$ . Then

$$t \cdot \mathbf{w}(m) = t \cdot \sum_{m' \neq m} 1_{P(m,m') > 0} + s \cdot \sum_{m' \neq m} 1_{P(m,m') = 0}. \quad (9)$$

The expression in Eq. (9) 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. (9) into an equivalent

expression that involves only entries in  $P$  above the diagonal. Condition 3 in Theorem 3, together with Eq. (3), 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,

$$\begin{aligned} t \cdot \mathbf{w}(m) = & t \cdot \left\{ \sum_{m' > m} 1_{P(m, m') > 0} + \sum_{m' < m} 1_{-P(m', m) > 0} \right\} + \\ & s \cdot \left\{ \sum_{m' > m} 1_{P(m, m') = 0} + \sum_{m' < m} 1_{P(m', m) = 0} \right\} \end{aligned} \quad (10)$$

Eq. (10) 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 in the bits of positivity in the first sum on the right-hand side of Eq. (10), by using the MPC sub-protocol described in Section 2.5. As for the bits of equality to zero in the second sum on the right-hand side of Eq. (10), the talliers can compute secret shares in them using the MPC sub-protocol described in Section 2.6. As the value of  $t \cdot \mathbf{w}(m)$  is a linear combination of those bits, the talliers can then use the secret shares in those bits and Eq. (10) in order to get secret shares in  $t \cdot \mathbf{w}(m)$ , 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 2.4) 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  $K$ th winning candidate. Namely, the overall number of comparisons in this final stage is  $\sum_{m=M-K}^{M-1} m = K \cdot (M - \frac{K+1}{2})$ , which 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 2, which is an implementation of Line 11 in Protocol 1. It assumes that the talliers hold  $D'$ -out-of- $D$  secret shares in  $P(m, m')$  for all  $1 \leq m < m' \leq M$ . Indeed, that computation has already taken place in Lines 9-10 of Protocol 1. Sub-protocol 2 starts with a computation of  $D'$ -out-of- $D$  shares in 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 in  $t \cdot \mathbf{w}(m)$  for each of the candidates, using Eq. (10); the  $D'$ -out-of- $D$  shares in  $t \cdot \mathbf{w}(m)$  are denoted  $\{w_d(m) : d \in [D]\}$ . Finally, using the secure comparison sub-protocol, they find the  $K$  winners (Lines 8-10).

---

**Sub-Protocol 2:** Computing the winners in COPELAND rule

---

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

```

1 forall  $1 \leq m < m' \leq M$  do
2   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
    $1_{P(m, m') > 0}$ ;
3   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
    $1_{-P(m, m') > 0}$ ;
4   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
    $1_{P(m, m') = 0}$ ;
5 forall  $d \in [D]$  do
6   forall  $m \in [M]$  do
7      $T_d$  computes
      $w_d(m) = t \cdot \{\sum_{m' > m} \sigma_d^+(m, m') + \sum_{m' < m} \sigma_d^-(m', m)\} +$ 
      $s \cdot \{\sum_{m' > m} \sigma_d^0(m, m') + \sum_{m' < m} \sigma_d^0(m', m)\};$ 
8 forall  $k \in [K] := \{1, \dots, K\}$  do
9   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  $k$ th elected candidate;
10  The talliers output the candidate that was found and remove him
   from  $\mathbf{C}$ ;
```

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

---

### 3.7 Coputing the winners in MAXIMIN rule

Fixing  $m \in [M]$ , the talliers need first to find the index  $m' \neq m$  which minimizes  $P(m, m')$ ; 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 — 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.

### 3.8 The protocol's security

The talliers hold  $D'$ -out-of- $D$  shares in 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  $D'$ , Eq. (1), the talliers cannot infer any information on any entry in any of the ballot matrices nor in the aggregated ballot matrix. They then use those shares in the following sub-protocols:

- (a) validating the legality of secret sharing (Section 3.5),
- (b) validating the legality of the ballots (Section 3.4), and
- (c) computing the final election results (Sub-protocol 2).

All of those sub-protocols are perfectly secure, as we proceed to explain. The protocol for validating the legality of secret sharing preserves perfect privacy, as shown in Lemma 5. The validation protocols of Section 3.4 are based on the arithmetic circuit construction of (Damgård and Nielsen, 2007) and (Chida et al., 2018), which was shown there to be secure. Finally, Sub-protocol 2 invokes the positivity testing sub-protocol (Lines 2-3), the sub-protocol that tests equality to zero (Line 4), and the secure comparison sub-protocol (Line 9), that we described in Sections 2.4–2.6. The secure comparison sub-protocol is perfectly secure, as was shown in (Nishide and Ohta, 2007). 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 arithmetic circuit construction of (Damgård and Nielsen, 2007) and (Chida et al., 2018), which was shown there to be secure.

In summary, Protocol 1 invokes five MPC sub-protocols, each of which is perfectly secure under the assumption of an honest majority. In view of the Modular Composition Theorem (Canetti, 2000), we arrive at the following

conclusion.

**Theorem 6.** *Under the assumption that the talliers are all semi-honest and that they have an honest majority, Protocol 1 is secure against malicious voters, it outputs the correct election results, and it fully preserves the voters’ privacy.*

### 3.9 End-to-end system

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, outside our MPC protocols.

Another requirement is the need to prevent attacks from 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) 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 receiver. 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.

Lastly, an important aspect of every voting system is *verifiability*. This property can be achieved by applying techniques such as the ones presented by Huber et al. (2022). Those techniques do not depend on the underlying voting rule, and they can be applied on top of our MPC protocols for computing the election results in a secure manner. We note, however, that Huber et al. (2022) point out a trade-off between full privacy and verifiability; specifically, for a system to be verifiable, the talliers must learn at least the aggregated votes.

## 4 Evaluation: Computational costs

Our goal herein is to establish the practicality of our protocol. Our protocol relies on expensive cryptographic sub-protocols — secure comparisons and secure multiplications. All other operations that the voters and talliers do (random number generation, and standard/non-secure additions and multiplications) have negligible costs in comparison to those of the cryptographic computations. In this section, we provide upper bounds for the overall cost of the cryptographic computations, in various election settings, in order to show that our protocol is viable and can be implemented in practical elections with very light overhead.

### 4.1 Parameters

Four parameters affect the performance of our protocol:

(a) **The size  $p$  of the underlying field.** We chose the prime  $p = 2^{31} - 1$  as the size of the underlying field, which is sufficiently large for our purposes (as discussed in Appendix I). Moreover, that specific setting of  $p$  serves us well also due to another reason:  $p = 2^{31} - 1$  is a Mersenne prime (namely, a prime of the form  $2^t - 1$  for some integer  $t$ ). Choosing such primes is advantageous, from a computational point of view, since multiplying two elements in such fields can be done without performing an expensive division (in case the multiplication result exceeds the modulus).

(b) **The number  $D$  of talliers.** We set that number to be  $D \in \{3, 5, 7, 9\}$ .

(c) **The number  $M$  of candidates.** Order-based rules require each voter to provide a full ordering of the entire set of candidates. Pairwise comparison is a common method for eliciting voter preferences when a full order is required. A sequence of comparative questions in the form of “Which of the following two candidates do you prefer?” are easier for the voter than a request for a complete order, see David (1963). Whether the voters are required to submit a full ranking of the set of  $M$  candidates, or they need to compare pairs of candidates (in which case roughly  $\log_2(M!)$  questions are needed in order to determine a full ordering of the  $M$  candidates), it is clear that such order-based rules are relevant only for a small number of candidates. In our evaluation, we considered  $M \in \{5, 10, 20\}$ .

(d) **The number  $N$  of voters.** That number affects only the time for validating the cast ballots. We provide here runtimes for validating batches of  $B$  ballots, for  $B \in \{500, 5000, 25000\}$ . Those runtimes should be multiplied

#layers	#multiplication gates per layer	$D = 3$	$D = 5$	$D = 7$	$D = 9$
20	50000	826	844	1058	1311
100	10000	842	989	1154	1410
1000	1000	1340	1704	1851	2243

Table 1: Runtimes (milliseconds) for computing  $10^6$  multiplication gates, spread evenly over 20, 100, and 1000 layers, as a function of the number  $D$  of talliers. The first two columns show the number of layers and the number of multiplication gates per layer in each setting.

by  $N/B$  in order to get the overall time for validating all incoming ballots.

## 4.2 The cost of batch validation of ballots

As discussed in Section 3.4, the batch validation of  $B$  ballots involves  $BM(M-1)/2$  simultaneous multiplications (for verifying condition 1) and  $M(M-1)/2$  consecutive rounds with  $B$  simultaneous multiplications in each (for verifying condition 4). We can perform the verification of both conditions in parallel by spreading the  $BM(M-1)/2$  simultaneous multiplications for verifying condition 1 over  $M(M-1)/2$  consecutive rounds with  $B$  simultaneous multiplications in each. Hence, the total workload would be  $M(M-1)/2$  rounds with  $2B$  simultaneous multiplications in each round.

Chida et al. (2018) report runtimes for performing secure multiplications. Their experiments were carried on Amazon AWS `m5.4xlarge` machines at N. Virginia over a network with bandwidth 9.6Gbps. They experimented over a larger field of size  $p' = 2^{61} - 1$  (which is also a Mersenne prime). Clearly, runtimes for our smaller prime,  $p = 2^{31} - 1$ , would be shorter; but since we are interested only in upper-bounding the computational overhead of our protocol, in order to establish its practicality, those numbers will suffice for our needs. They experimented with a circuit that consists of one million multiplication gates that are evenly spread over  $\{20, 100, 1000\}$  layers; hence, in each layer there are  $\{5 \cdot 10^4, 10^4, 10^3\}$  multiplication gates, respectively. The reported runtimes as a function of  $D$ , the number of talliers, are shown in Table 1.

We begin by computing the needed runtimes for performing batch validations when  $M = 20$ . We exemplify the computation in two cases: one in which the batch size is  $B = 500$  and another in which the batch size is

	$B = 500$	$B = 5000$	$B = 25000$
$M = 5$	27/34/37/45	17/20/23/28	16/17/21/26
$M = 10$	121/153/167/202	76/89/104/127	74/76/95/118
$M = 20$	510/648/704/852	320/376/439/536	314/321/402/498

Table 2: Runtimes (seconds) for validating 1 million ballots in order-based rules, as a function of the number of candidates  $M$ , the batch size  $B$ , and the number of talliers  $D$ . The table’s entry relating to  $M$  and  $B$  shows the validation runtimes for  $D = 3/5/7/9$ .

increased 50-fold, i.e.,  $B = 25000$ .

To validate  $B = 500$  ballots when  $M = 20$  it is necessary to perform  $M(M - 1)/2 = 190$  rounds of  $2B = 1000$  simultaneous multiplications in each. The runtimes for such a computation can be inferred from the third row in Table 1, if we multiply the runtimes shown there by a factor of  $\frac{190}{1000}$ . That is, the batch validation of  $B = 500$  ballots would take 255/324/352/426 mili-seconds when  $D = 3/5/7/9$ . Therefore, to validate a million ballots, it would take 510/648/704/852 seconds.

Those runtimes may be improved by enlarging the batch size. The time to validate a batch of  $B = 25000$  ballots, when  $M = 20$ , can be inferred from the first row in Table 1, if we multiply the runtimes shown there by a factor of  $\frac{190}{20}$ . That is, the batch validation of  $B = 25000$  ballots would take 7.847/8.018/10.051/12.454 seconds when  $D = 3/5/7/9$ . Therefore, validating a million ballots, would take 314/321/402/498 seconds.

In general, it is not hard to see that the runtimes for validating one million ballots in batches of size  $B = 500/5000/25000$  can be read from Table 1 by multiplying the times reported there in the third/second/first row by a factor of  $M(M - 1)$ . Table 2 includes runtimes for validating 1 million ballots in batches of  $B \in \{500, 5000, 25000\}$  ballots when  $M \in \{5, 10, 20\}$ , for  $D = 3/5/7/9$ .

Elections usually span a long time (typically at least 1 day) and the batch validation of ballots can take place along that period whenever a number of  $B$  new ballots were received. Hence, the above analysis shows that the runtimes for validating incoming ballots are very realistic and are not expected to slow down the election process.



### 4.3 The cost of computing final election results

Sub-protocol 2 computes the final election results in the COPELAND rule. It requires  $M(M - 1)$  invocations of the secure positivity test (Section 2.5),  $M(M - 1)/2$  invocations of the equality to zero test (Section 2.6), and finally  $K \cdot (M - \frac{K+1}{2}) \leq M(M - 1)/2$  secure comparisons (Section 2.4). As discussed in Sections 2.5 and 2.6, the costs of the MPC computations to determine positivity and equality to zero are upper bounded by the cost of a secure comparison. Hence, the cost of Sub-protocol 2 can be upper bounded by  $4M(M - 1)$  secure comparisons.

To evaluate the runtime of performing the secure comparison sub-protocol we ran it on Amazon AWS m5.4xlarge machines at N. Virginia over a network with bandwidth 9.6Gbps. We performed our evaluation with  $D \in \{3, 5, 7, 9\}$  talliers. The measured runtimes are given in Table 3.

Number of talliers	$D = 3$	$D = 5$	$D = 7$	$D = 9$
Time (msecs) to compute a secure comparison	9.07	9.54	9.64	15.0

Table 3: Runtimes (milliseconds) for a secure comparison sub-protocol with a varying number of talliers.

As can be seen, the implied runtimes are negligible. For example, when  $M = 5$ , the runtime of this stage is upper bounded by 1.2 seconds, when using the highest number of talliers,  $D = 9$ , while for  $M = 20$  it is upper bounded by 22.8 seconds. The runtimes in the case of MAXIMIN are even smaller, since then the number of secure comparisons is bounded only by  $1.5 \cdot M^2$  (see Section 3.7). In summary, it is possible to achieve perfect ballot secrecy, as our protocol offers, at a very small computational price.

## 5 Related work

Secure e-voting can be approached using various cryptographic techniques. The earliest suggestion is that of Chaum (1981), 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 (1995); Adida (2008); Boneh and Golle

(2002); Jakobsson et al. (2002); Lee et al. (2003); Neff (2001). However, while such systems preserve 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 (Dery et al., 2021).

One of the approaches towards achieving tally-hiding privacy, and not just anonymity, is by employing homomorphic encryption. 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 (1986). 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. (Cramer et al., 1997; Damgård et al., 2010; Hevia and Kiwi, 2004; Yang et al., 2018; Fan et al., 2020; Rezaeibagha et al., 2019; Priya et al., 2018).

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. (2018) considered the MAJORITY JUDGMENT (MJ) voting rule (Balinski and Laraki, 2007). 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. (2015) suggested to use secret sharing for the tallying process in PLURALITY voting (Brandt et al., 2016). 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. (2020) 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. (2021) 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 began looking at order-based voting rules. Haines et al. (2019) proposed a solution for the order-based SCHULZE’s rule (Schulze, 2011). 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. (2021) 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. (2022) considered the SINGLE TRANSFERABLE VOTE (STV) rule, which is a multi-stage rule, as well as SCHULZE’s rule. Even though their method is much more efficient than the one in Hertel et al. (2021) 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 solution even for very large democracies. For example, our protocols can handle COPELAND rule computations over 1 million voters and 20 candidates with a runtime of roughly 500 seconds (see Table 2). This is achieved mainly by our novel idea of distributed tallying. However, applying distributed tallying for tackling more complex rules such as SCHULZE’s rule requires further research and is left for future work.

## 6 Conclusion

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. Secure voting systems that rely on fully trusted talliers (that is, talliers who receive the actual ballots from the voters) assume 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. 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 of 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 COPELAND and MAXIMIN. In the Appendix we present our solution for KEMENY-YOUNG and MODAL-RANKING. 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 democracies of any size.

## References

- Ben Adida. 2008. Helios: Web-based Open-Audit Voting.. In *USENIX security symposium*, Vol. 17. 335–348.
- 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.
- J.C. Benaloh. 1986. Secret sharing homomorphisms: Keeping shares of a secret secret. In *CRYPTO*. 251–260.
- G.R. Blakley. 1979. Safeguarding Cryptographic Keys. In *International Workshop on Managing Requirements Knowledge*. 48: 313–317.
- Dan Boneh and Philippe Golle. 2002. Almost entirely correct mixing with applications to voting. In *Proceedings of the 9th ACM conference on Computer and communications security*. 68–77.
- Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice*. Cambridge University Press.
- Felix Brandt and Tuomas Sandholm. 2005. Decentralized voting with unconditional privacy. In *AAMAS*. 357–364.
- Ernest F. Brickell. 1989. Some Ideal Secret Sharing Schemes. In *EUROCRYPT*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.). 468–475.
- Sébastien Canard, David Pointcheval, Quentin Santos, and Jacques Traoré. 2018. Practical strategy-resistant privacy-preserving elections. In *European Symposium on Research in Computer Security*. Springer, 331–349.
- Ran Canetti. 2000. Security and Composition of Multiparty Cryptographic Protocols. *J. of Cryptology* 13 (2000), 143–202.
- Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. 2014. Modal ranking: A uniquely robust voting rule. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- David Chaum. 1988. Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA. In *EUROCRYPT*. 177–182.

- David L Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.
- 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.
- 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*.
- 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*. Springer, 631–652.
- Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. 1997. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*. 103–118.
- 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, 6 (2010), 371–385.
- Ivan Damgård and Jesper Buus Nielsen. 2007. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*. 572–590.
- Herbert Aron David. 1963. *The method of paired comparisons*. Vol. 12. London.
- 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.
- 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, 1 (2009), 275–341.
- 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.

- Oriol Farràs, Jaume Martí-Farré, and Carles Padró. 2012. Ideal Multipartite Secret Sharing Schemes. *J. Cryptol.* 25 (2012), 434–463.
- Thomas Haines, Dirk Pattinson, and Mukesh Tiwari. 2019. Verifiable homomorphic tallying for the Schulze vote counting scheme. In *Working Conference on Verified Software: Theories, Tools, and Experiments*. Springer, 36–53.
- 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.
- Alejandro Hevia and Marcos Kiwi. 2004. Electronic jury voting protocols. *Theoretical Computer Science* 321, 1 (2004), 73–94.
- Nicolas Huber, Ralf Küsters, Toomas Krips, Julian Liedtke, Johannes Müller, Daniel Rausch, Pascal Reisert, and Andreas Vogt. 2022. Kryvos: Publicly tally-hiding verifiable e-voting. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1443–1457.
- Markus Jakobsson, Ari Juels, and Ronald L Rivest. 2002. Making mix nets robust for electronic voting by randomized partial checking.. In *USENIX security symposium*. San Francisco, USA, 339–353.
- John G Kemeny. 1959. Mathematics without numbers. *Daedalus* 88, 4 (1959), 577–591.
- Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. 2020. Ordinos: A verifiable tally-hiding e-voting system. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 216–235.
- 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*. Springer, 245–258.
- 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).

- C Andrew Neff. 2001. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*. 116–125.
- Takashi Nishide and Kazuo Ohta. 2007. Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In *PKC*. 343–360.
- 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*. Springer, 811–819.
- Fatemeh Rezaeibagha, Yi Mu, Shiwei Zhang, and Xiaofen Wang. 2019. Provably secure (broadcast) homomorphic signcryption. *International Journal of Foundations of Computer Science* 30, 04 (2019), 511–529.
- Kazue Sako and Joe Kilian. 1995. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 393–403.
- Markus Schulze. 2011. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social choice and Welfare* 36, 2 (2011), 267–303.
- Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22 (1979), 612–613.
- Paul B Simpson. 1969. On defining areas of voter choice: Professor Tullock on stable voting. *The Quarterly Journal of Economics* (1969), 478–490.
- Alan Szeponiec and Bart Preneel. 2015. New techniques for electronic voting. In *{USENIX} Journal of Election Technology and Systems ({JETS})*.
- Tamir Tassa and Nira Dyn. 2006. Multipartite Secret Sharing by Bivariate Interpolation. In *ICALP*. 288–299.
- 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.



- A.C. Yao. 1982. Protocols for secure computation. In *FOCS*. 160–164.
- Peyton Young. 1995. Optimal voting rules. *Journal of Economic Perspectives* 9, 1 (1995), 51–64.

## A Emulating multiplication gates

Here we describe how the generic secret-sharing-based protocol of Damgård and Nielsen (2007) emulates multiplication gates. That computation requires the interacting parties to generate shares in a random field element  $r$ , such that  $r$  distributes uniformly on  $\mathbb{Z}_p$  and it remains unknown to all parties. We begin by describing a manner in which this latter task can be carried out.

To generate shares in a random field element, each party  $T_d$ ,  $d \in [D]$ , generates a uniformly random field value  $\rho_d$  and performs a  $D'$ -out-of- $D$  sharing of it among  $T_1, \dots, T_D$ . At the completion of this stage, each  $T_d$  adds up all the  $D$  shares that he received and gets a value that we denote by  $r_d$ . It is easy to see that  $\{r_1, \dots, r_D\}$  is a  $D'$ -out-of- $D$  sharing of the random value  $\rho = \sum_{d \in [D]} \rho_d$ . Clearly,  $\rho$  is a uniformly random field element, as it is a sum of uniformly random independent field elements.

Now we turn to explain the processing of multiplication gates. Let  $\{u_1, \dots, u_D\}$  be  $D'$ -out-of- $D$  shares in  $u$ , which were generated by a polynomial  $f(\cdot)$  of degree  $D' - 1$ , and  $\{v_1, \dots, v_D\}$  be  $D'$ -out-of- $D$  shares in  $v$ , which were generated by a polynomial  $g(\cdot)$  also of degree  $D' - 1$ . The party  $T_d$  holds  $u_d$  and  $v_d$ ,  $d \in [D]$ . The goal is to let the parties have  $D'$ -out-of- $D$  shares in  $w = u \cdot v$ .

First,  $T_d$  computes  $w_d = u_d \cdot v_d$ ,  $d \in [D]$ . Those values are point values of the polynomial  $f(\cdot)g(\cdot)$ , which is a polynomial of degree  $2D' - 2$ . Hence,  $\{w_1, \dots, w_D\}$  is a  $(2D' - 1)$ -out-of- $D$  sharing of  $w$ . Note that as  $D' := \lfloor (D + 1)/2 \rfloor$ , then  $2D' - 1 \leq D$ ; therefore, the  $D$  parties have a sufficient number of shares in order to recover  $w$ . However, our goal is to obtain a  $D'$ -out-of- $D$  sharing of  $w$ , namely a set of shares in  $w$ , of which any selection of only  $D'$  shares can be used to reconstruct  $w$ . Hence, we proceed to describe a manner in which the parties can translate this  $(2D' - 1)$ -out-of- $D$  sharing of  $w$  into a  $D'$ -out-of- $D$  sharing of  $w$ . To do that, the parties generate two sharings of the same uniformly random (and unknown) field element  $R$ : a  $D'$ -out-of- $D$  sharing, denoted  $\{r_1, \dots, r_D\}$ , and a  $(2D' - 1)$ -out-of- $D$  sharing, denoted  $\{R_1, \dots, R_D\}$ . Next, each  $T_d$  computes  $\tilde{w}_d = w_d + R_d$  and sends the result to  $T_1$ . Since  $\{\tilde{w}_1, \dots, \tilde{w}_D\}$  is a  $(2D' - 1)$ -out-of- $D$  sharing of  $w + R$ ,  $T_1$  can use any  $2D' - 1$  of those shares in order to reconstruct  $\tilde{w} := w + R$ .  $T_1$  broadcasts that value to all parties. Consequently, each  $T_d$  computes  $\hat{w}_d = \tilde{w} - r_d$ ,  $d \in [D]$ . Since  $\tilde{w}$  is a constant and  $r_d$  is a  $D'$ -out-of- $D$  share in  $R$ , then  $\hat{w}_d$  is a  $D'$ -out-of- $D$  share in  $\tilde{w} - R = w + R - R = w$ , as needed. This procedure is perfectly secure since  $\tilde{w} = w + R$  reveals no information

on  $w$  because  $R$  is a uniformly random field element that is unknown to the parties.

## B Proof of Lemma 1

If  $q < \frac{p}{2}$  then  $2q < p$ . Hence,  $2q \bmod p = 2q$  (there is no modular reduction), and therefore, as  $2q$  is even, its LSB is 0. On the other hand, if  $q > \frac{p}{2}$  then  $2q > p$ . Hence,  $2q \bmod p = 2q - p$ . Since that number is odd, its LSB is 1.  $\square$

## C Proof of Lemma 2

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

## D Proof of Theorem 3

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

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

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  $m$ th column 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 column, equals  $2k - M - 1$ . 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  $m$ th column 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 column 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 column equals  $2k(m) - M - 1$ . As, by condition 4, all  $Q_m$  values are distinct, then  $k(m) \neq k(m')$  when  $m \neq m'$ . Stated otherwise, the sequence  $\mathbf{k} := (k(1), \dots, k(M))$  is a permutation of  $[M]$ . Let  $\mathbf{j} := (j_1, \dots, j_M)$  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.  $\square$

## E Proof of Theorem 4

The proof of Theorem 4 is similar to that of Theorem 3 and thus omitted.

## F Proof of Lemma 5

Assume that  $V$  is honest. Then there exists a polynomial  $g$  of degree at most  $D' - 1$  such that  $s_d = g(d)$ ,  $d \in [D]$ . Define

$$G := g + \sum_{j \in [D]} g_j, \quad (11)$$

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

$$G(d) = g(d) + \sum_{j \in [D]} g_j(d) = s_d + \sum_{j \in [D]} r_{j,d} = \hat{s}_d, \quad d \in [D]. \quad (12)$$

Hence, the interpolating polynomial  $f$  that the talliers compute in Step 4 coincides with  $G$ . As  $G$  is a sum of polynomials of degree  $D' - 1$  at most, the validation in Step 5 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 in Step 1,  $j \in [D]$ . 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 in one of his ballot matrix entries. Namely,  $V$  had distributed shares  $\{s_1, \dots, s_D\}$  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 Step 4 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. Hence, the verification procedure is correct and provides perfect privacy.  $\square$

## G The Kemeny-Young rule

Here we consider the KEMENY-YOUNG rule (Kemeny, 1959; Young, 1995) and discuss its secure implementation. In this rule, the ballot of each voter is a ranking of all candidates, where ties are allowed. Hence, the ballot of  $V_n$ ,  $n \in [N]$ , may be described by an  $M$ -dimensional array,  $R_n$ , where the  $m$ th entry in that array,  $R_n(m)$ ,  $m \in [M]$ , is a number in  $[M]$  that equals the rank of  $C_m$  in  $V_n$ 's preference list.

For example, assume that there are four candidates: Alice ( $C_1$ ), Bob ( $C_2$ ), Carol ( $C_3$ ) and David ( $C_4$ ). Then if  $V_n$  ranks Carol as her top candidate, Bob as her second choice, and either Alice or David as her last choices, then her ballot would be the vector  $R_n = (3, 2, 1, 3)$ . This vector ballot is then translated into a matrix  $P_n$  of dimensions  $M \times M$ , where  $P_n(m, \ell) = 1$  if  $C_m$  is ranked strictly higher than  $C_\ell$  in  $R_n$ , and  $P_n(m, \ell) = 0$  otherwise. In the above example,

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

The talliers aggregate the incoming ballots by computing  $P = \sum_{n \in [N]} P_n$ . For every  $1 \leq m \neq \ell \leq N$ ,  $P(m, \ell)$  equals the number of voters who ranked  $C_m$  strictly higher than  $C_\ell$ . The matrix  $P$  induces a score for each of the possible  $M!$  rankings over  $\mathbf{C} = \{C_1, \dots, C_M\}$ . Let  $\rho = (\sigma_1, \dots, \sigma_M)$ , where  $(\sigma_1, \dots, \sigma_M)$  is a permutation of  $\{1, \dots, M\}$ , be such a ranking. Here, for each  $m \in [M]$ ,  $\sigma_m$  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  $\rho$  is defined as

$$w(\rho) = \sum_{\ell, m \in [M]: \sigma_m < \sigma_\ell} P(m, \ell); \quad (13)$$

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

We now turn to the secure implementation of the KEMENY-YOUNG rule. 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. Secret sharing is applied on each of the  $M^2 - M$  non-diagonal entries of  $P_n$ , since the diagonal entries are always zero.

To validate each in-coming ballot matrix, the talliers only need to check that all entries are in  $\{0, 1\}$  and that for every  $1 \leq m < \ell \leq M$ ,  $P_n(m, \ell) + P_n(\ell, m) \in \{0, 1\}$ . Indeed, the sum of two opposing entries,  $P_n(m, \ell)$  and  $P_n(\ell, m)$ , will always equal 1 (if one of  $C_m$  and  $C_\ell$  is ranked higher than the other) or 0 (if both are in a tie). The validation of such conditions goes along the lines that we described in Section 3.4.

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

To compute secret shares in the score of each possible ranking  $\rho$ , the talliers need only to perform summation according to Eq. (13). 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 2.4.

## H Modal ranking

In MODAL RANKING (Caragiannis et al., 2014), 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. (2016)) 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. (2021).

## I 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  $\mathbb{Z}_p$ .

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. (3), 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. (9), 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. (7). 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  $\alpha$  in COPELAND rule, Eq. (4)), are typically much smaller than  $N$ , 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.