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

Tamir Tassa
The Open University
Ra'anana, Israel
tamirta@openu.ac.il

Lihi Dery
Ariel Cyber Innovation Center
Ariel University
Ariel, Israel
lihid@ariel.ac.il

ABSTRACT

Electronic voting systems are essential for holding virtual elections. One of the main challenges in such elections is to secure the voting process: namely, to certify that the computed results are consistent with the cast ballots, and that the privacy of the voters 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, which is 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

ACM Reference Format:
Tamir Tassa and Lihi Dery. 2022. Towards Secure Virtual Elections: Multiparty Computation of Order Based Voting Rules. In *Appears at the 4th Games, Agents, and Incentives Workshop (GAIW 2022). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., Auckland, New Zealand, May 2022*, IFAAMAS, 12 pages.

1 INTRODUCTION

Secure e-voting systems are required for holding elections virtually. Virtual elections may increase participation, reduce costs, and increase sustainability. In the past two years, due to the COVID-19 pandemic and the need for social distancing, secure e-voting platforms have become even more essential.

The usual meaning of voter privacy is that the voters remain anonymous. Namely, even though the ballots are revealed (as is the case when opening the ballot box at the end of an election day), no ballot can be traced back to the voter who cast it. Herein we propose a protocol that offers perfect ballot secrecy, or full privacy [9], 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 perfect secrecy may increase the voters' confidence and, consequently, encourage them 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, each voter assigns a score to each of the candidates. In this study we focus on *order-based* voting rules, where each voter submits a ranking of the candidates [7].

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

2 RELATED WORK

Secure e-voting can be approached using various cryptographic techniques. The earliest suggestion is that of Chaum [10], who suggested to use 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. Others followed and improved this model (e.g. [1, 5, 21, 22, 24, 28]). However, while this system preserves 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 [15].

Homomorphic encryption enable 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 here 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 who decrypts the received ciphertext recovers the aggregation of the ballots, but not the ballots themselves. Secure voting protocols that are based on homomorphic encryption were presented in e.g. [12, 13, 20, 26, 27, 30].

While most studies on secure voting offered protocols for securing the voting process, only few studies considered the question of private execution of the computation that the underlying voting rule dictates. Canard et al. [8] considered the Majority Judgment

Appears at the 4th Games, Agents, and Incentives Workshop (GAIW 2022). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., Abramowitz, Ceppi, Dickerson, Hosseini, Lev, Mattei, Zick (Chairs), May 2022, Auckland, New Zealand. © 2022 Copyright held by the owner/author(s). …$ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

(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. [23] suggested to use secret sharing for the tallying process in Plurality voting. 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. Yang et al. [30] suggested using homomorphic encryption specifically for approval voting [6]. Lastly, Dery et al. [15] offered a solution based on multiparty computation in order to securely determine the winners in elections governed by score-based voting rules.

To the best of our knowledge, no study so far addressed the question of securely computing election results where the governing voting rule is order-based. We undertake this challenge herein.

3 CRYPTOGRAPHIC PRELIMINARIES

Our protocol relies heavily on cryptographic machinery. Herein we provide a brief introduction to secret sharing, secure multiparty computation (MPC), and solutions to specific problems of MPC that are used in our protocol: secure comparisons, secure testing of positivity, and secure testing of equality to zero.

3.1 Secret sharing

Secret sharing methods [29] enable distributing a secret among a group of participants. Each participant is given a random share of the secret so that: (a) the secret can be reconstructed only by combining the shares given to specific *authorized* subsets of participants, and (b) combinations of shares belonging to unauthorized subsets of participants reveal zero information on the underlying secret.

The notion of secret sharing was introduced, independently, by Shamir [29] and Blakley [4], for the case of threshold secret sharing. Let D be the number of participants and let $D' \leq D$ be some threshold. Then the authorized subsets in Shamir's and in Blakley's schemes 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 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, $s_d = g(d)$, $d \in [D] := \{1, \ldots, D\}$, where s_d is the share given to the dth participant, $d \in [D]$.
- $\mathsf{Reconstruct}_{D'}(s_1, \ldots, s_D)$. 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 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 s.

Our protocol involves a set of third parties, $\{T_1, \ldots, T_D\}$, which are called *talliers*. In the protocol, each voter creates shares of his private ballot and distributes them to the D talliers. As those ballots are matrices (see the definitions in Section 4.1), the secret sharing is carried out for each entry independently, so that each of the tallier receives a share matrix in each ballot matrix. The talliers then verify the legality of the cast ballots and, at the end the election period, compute the desired election results. Those computations are executed based only on the shares that the talliers had received, i.e., without actually recovering the private ballots. Such a computation, that depends on secret inputs that cannot be disclosed, is called *secure multiparty computation*; we elaborate on that topic in Section 3.2 below.

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 collude in order to recover the secret ballots. If the set of the 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 attempt to recover the secret ballots from the shares that they hold, they will not be able to extract any information on the ballots. The secret sharing mechanism prevents such unauthorized subsets from inferring any information on the shared secrets from the shares that they hold.

Higher values of D (and consequently of $D' := \lfloor (D+1)/2 \rfloor$) will imply greater security against coalitions of corrupted talliers, but at the same time they will also imply higher computational and communication costs.

3.2 Secure multiparty computation

As described above, the talliers hold shares in the voters' ballots and perform computations on those shares. First, they need to verify that the shares for each cast ballot correspond to a legal ballot (legal ballots are characterized in Section 4.2). Then, they need to compute the identity of the K winning candidates, as dictated by the rule. Those two 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 those computations on the distributed shares, without revealing the shares to each other. As shown in Section 4.2, both of those computations boil down to arithmetic computations (namely, to computations of additions and multiplications of secret-shared values).

To do so, the talliers execute sub-protocols of secure multiparty computation (MPC) [31]. An MPC protocol allows a set of parties, $T_1, \ldots, T_D$, to compute any function f over private inputs that they hold, $x_1, \ldots, x_D$, where x_d is known only to T_d, $d \in [D]$, so that at the end of the protocol everyone learns $f(x_1, \ldots, x_D)$, but nothing beyond that value and what can be naturally inferred from it. In cases where the desired function f may be expressed as an

arithmetic function of the inputs, then one may represent f by an arithmetic circuit C such that for every set of inputs, $x_1, \ldots, x_D$, the output of the circuit, $C(x_1, \ldots, x_D)$, equals $f(x_1, \ldots, x_D)$.

The circuit has several input wires and a single output wire. Each input wire is fed with a single secret value by one of the parties. The output wire determines a single value that is revealed to all parties. Between the input and output wires there are multiple layers of arithmetic gates that connect them. An arithmetic gate can be either addition or multiplication. Each gate is fed by exactly two input wires, and it produces one output wire such that the output of a gate in layer k can be given as input to multiple gates in layer $k+1$. For illustration, consider the arithmetic function $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 \cdot x_3 \cdot (x_2 + x_3)$. The circuit C in Figure 1 evaluates that function. It consists of three layers. The first layer has two multiplication gates that compute $x_1 \cdot x_2$ and $x_2 \cdot x_3$, and one addition gate that computes $x_2 + x_3$. The second layer has a single multiplication gate for computing $x_2 \cdot x_3 \cdot (x_2 + x_3)$. Finally, the third and last layer has a single addition gate that issues the desired output.

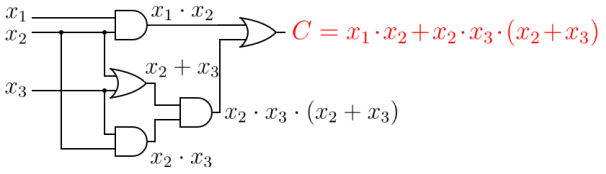

Figure 1: An arithmetic circuit C that realizes the function $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 \cdot x_3 \cdot (x_2 + x_3)$.

In a secret-sharing-based MPC protocol, all values along all wires are secret shared among the D parties (as described in Section 3.1); in particular, they are never reconstructed and thus remain secret. Only the value on the final output wire is reconstructed: when reaching the output gate, each party broadcasts the corresponding share that he[1] holds in the output wire, and subsequently everyone can use the broadcast shares in order to reconstruct the output.

The computational challenge in this context is how to emulate the operation of an arithmetic circuit when operating on shared values. As an arithmetic circuit consists of two types of gates – addition and multiplication, the challenge is as follows: given D'-out-of-D shares in two secret values $u, v \in \mathbf{Z}_p$, how can the parties compute D'-out-of-D shares in $u + v$ and in $u \cdot v$, without reconstructing any of those values. In the circuit shown in Figure 1, the parties have D'-out-of-D shares in each of the input values x_1, x_2, x_3. With those shares they proceed to emulate the circuit layer by layer, by computing proper D'-out-of-D shares in the output wires of the three gates in the first layer, then proceeding to computing D'-out-of-D shares in the output of the multiplication gate in the second layer, using the already computed shares in the output wires of two of the gates in the first layer, and finally computing D'-out-of-D shares in the output wire using the computed shares in the output wires of the multiplication gate in the second layer and the first multiplication gate in the first layer.

[1]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 both genders, or talliers, that are typically (genderless) servers.

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. Emulating addition gates is easy: the linearity of secret sharing implies that for any two public field elements $a, b \in \mathbf{Z}_p$, the values $au_d + bv_d$, $d \in [D]$, are proper D'-out-of-D shares in $au + bv$. In particular, each party may compute his share in the output wire from his shares in the input wires without any interaction with the other parties.

Emulating multiplication gates is trickier. Here, in order to compute proper D'-out-of-D shares in the output wire $u \cdot v$, the parties need to interact, and the required computation is more involved. In our protocol we use the construction proposed by Damgård and Nielsen [14], enhanced by a work by Chida et al. [11] 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 it in Appendix A.

3.3 Secure Comparison

When computing election results, it is essential to repeatedly compare scores of two candidates in order to determine which is larger. In our protocol, those scores are kept hidden from all parties, but the talliers hold secret shares in them. Of course, the talliers could use those shares in order to recover the scores and then compare them. However, for the sake of achieving perfect ballot secrecy, our goal is to determine which of two given scores is larger without explicitly recovering those scores. To that end, we consider the problem of secure comparison, which is defined below.

Assume that the D parties, $T_1, \ldots, T_D$, 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 $\mathbf{Z}_p$. The parties wish to compute D'-out-of-D secret shares in the binary variable, denoted $1_{a<b}$, which equals 1 if $a < b$ and 0 otherwise[2], without learning any other information on a and b. A protocol that does that is called *secure comparison*.

Nishide and Ohta [25] presented a method for secure comparison that is based on the following simple observation. Let us denote the bits (i.e., binary variables) $1_{a<\frac{p}{2}}$, $1_{b<\frac{p}{2}}$, $1_{[(a-b) \mod p]<\frac{p}{2}}$, and $1_{a<b}$ by w, x, y, z, respectively. Then

$$z = w\bar{x} \vee \bar{w}\bar{x}\bar{y} \vee wx\bar{y}\,. \tag{1}$$

The Boolean expression in Eq. (1) can be translated to an equivalent arithmetic expression:

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

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 alternative expression efficiently computable is the fact that in the three comparison bits, $w = 1_{a<\frac{p}{2}}$, $x = 1_{b<\frac{p}{2}}$, and $y = 1_{[(a-b) \mod p]<\frac{p}{2}}$, the right-hand side is $\frac{p}{2}$, as we proceed to explain.

[2]Hereinafter, for any predicate Π, we let 1_Π denote the binary variable that equals 1 if and only if the predicate Π holds.

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

The proof can be found in Appendix B. In view of Lemma 1, the parties 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 party T_d will translate the share he holds in a to a share in $2a$ by multiplying the share by 2; then, all parties will use their shares in $2a$ in order to compute shares in the LSB of $2a$. The reader is referred to Nishide and Ohta [25] for the details of that last step in the computation.

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 $\ell = \log_2(p)$. Hence, computing shares in w, x, and y requires 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 279ℓ+5 multiplications.

3.4 Secure testing of positivity and equality to zero

Let x be an integer in the range $[-N, N]$. Assume that $T_1, \ldots, T_D$ hold D'-out-of-D shares in x, where the underlying field is $\mathbf{Z}_p$, and $p > 2N$. They wish to obtain D'-out-of-D shares in the bits $1_{x>0}$ and $1_{x=0}$, without learning any further information on x.

For the first task, we state the following lemma, the proof of which is given in Appendix C.

Lemma 2. *Under the above assumptions, $x > 0$ if and only if the LSB of $(-2x \mod p)$ is 1.*

Hence, the talliers need only to multiply their shares in x by -2 and then compute the LSB of shared value. Therefore, the computational cost of that task is 13 rounds of communication and $93\ell + 1$ multiplications.

For the task of computing shares in the bit $1_{x=0}$, we recall Fermat's little theorem that states that if $x \in \mathbf{Z}_p \setminus \{0\}$ then $x^{p-1} = 1 \bmod p$. Hence, $1_{x\neq 0} = \left(x^{p-1} \mod p\right)$. Therefore, shares in the bit $1_{x\neq 0}$ can be obtained by simply computing $x^{p-1} \bmod p$. The latter computation can be carried out by the square-and-multiply algorithm with up to 2ℓ 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 $1_{x=0}$.

4 THE METHOD: 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.

4.1 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 $\mathbf{C}$, or the K leading candidates in that ranking, for some pre-determined $K < M$.

We proceed to define the two order-based rules for which we devise a secure MPC protocol in this paper. In the full version of this work, we also address Kemeny-Young and Modal Ranking.

• 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,

$$P = \sum_{n=1}^{N} P_n \,, \tag{3}$$

induces the following score for each candidate:

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

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$^{\frac{1}{2}}$ [18].

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

We hereinafter refer to the matrix P_n as V_n's *ballot matrix*, and to P as the aggregated ballot matrix.

4.2 Characterization of legal ballot matrices

Here we characterize the ballot matrices in each of the two order-based rules that we consider. This 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 to sabotage the elections.

Theorem 3. *An $M \times M$ matrix Q is a valid ballot under the* Copeland *rule if and only if 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 $\{Q_m : m \in [M]\}$, where $Q_m := \sum_{m' \in [M]} Q(m', m)$, consists of M distinct values.

The proof of Theorem 3 is given in Appendix D.

Theorem 4. *An $M \times M$ matrix Q is a valid ballot under the* Maximin *rule if and only if 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 $\{Q_m : m \in [M]\}$, where $Q_m := \sum_{m' \in [M]} Q(m', m)$, consists of M distinct values.

The proof of Theorem 4 is similar to that of Theorem 3 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, $\Gamma(Q) := (Q(m,m') : 1 \leq m < m' \leq M)$. 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.

4.3 A secure voting protocol

Protocol 1 is 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: a voting phase and a tallying phase. We first provide a bird's-eye view of those two phases.

In Phase 1 (Lines 1-8), the voting 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 3.1 (Lines 3-7). Following that, the talliers jointly verify the legality of the shared ballot (Line 8). In Phase 2 (Lines 9-11), 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 4.2 (Lines 3-5). Then, V_n sends each tallier T_d 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).

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, see Section 4.4. 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 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. (3), 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 in $P_n(m,m')$ 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 in $P(m,m') = \sum_{n \in [N]} P_n(m,m')$.

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. How can the talliers do that when none of them actually holds the matrix P? We devote our discussion in Sections 4.5 and 4.6 to that interesting computational challenge.

4.4 Verifying the legality of the cast ballots

Here we provide a short overview of how the talliers may validate a cast ballot matrix, Q, using the shares that were dealt to them in its upper triangle, $\Gamma(Q)$, without recovering the ballot and without breaching the voter's privacy. The full discussion is given in Appendix E.

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 $a_{n,m,m',j} \in \mathbf{Z}_p$, $1 \leq j \leq D'-1$;
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 $T_1, \ldots, T_D$ find the indices of the K winners and publish them;

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

The validation is based on the characterizations of legal ballot matrices as provided in Theorems 3 and 4 for Copeland and Maximin, respectively. Since the talliers get shares only in $\Gamma(Q)$, they only need to verify conditions 1 and 4.

To validate condition 1 in Theorem 3 for Copeland, they need to verify that each entry in the shared $\Gamma(Q)$ is either 1 or -1. A shared scalar x is in $\{-1, 1\}$ iff $(x+1) \cdot (x-1) = 0$. Hence, the talliers input their shares in x to an arithmetic circuit that outputs the product $(x+1) \cdot (x-1)$. If the output of such a circuit is zero for each of the $M(M-1)/2$ entries of $\Gamma(Q)$, then $\Gamma(Q)$ satisfies condition 1 in Theorem 3. The verification of condition 1 in Theorem 4 for Maximin goes along the same lines, where the computed circuit outputs $x \cdot (x-1) = 0$., in ordet to verify that $x \in \{0, 1\}$.

To validate condition 4, the talliers compute shares in the M values Q_m, $m \in [M]$, and then compute the product $F(Q) := \prod_{1 \leq m' < m \leq M} (Q_m - Q_{m'})$. Condition 4 is satisfied iff $F(Q) \neq 0$.

The above outlined procedure reveals to the talliers nothing beyond the legality of the cast ballots. Hence, the voters may be assured that their privacy is fully preserved, see Appendix E.

Verifying condition 1 can be performed in parallel for all $M(M-1)/2$ entries in a given ballot. 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.

4.5 An MPC computation of the winners in the Copeland rule

The parameter α in Eq. (4) is always a rational number; typical settings of α are 0, 1, or $\frac{1}{2}$ [18]. Assume that $\alpha = \frac{s}{t}$ for some

integers s and t. Then

$$t \cdot \mathbf{w}(m) = t \cdot \sum_{m'} 1_{P(m,m')>0} + s \cdot \sum_{m'} 1_{P(m,m')=0}\,, \qquad (5)$$

where both summations are over $m' \in [M] \setminus \{m\}$. The expression in Eq. (5) 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, $1 \le m < m' \le M$ (see Lines 9-10 in Protocol 1). Hence, we first translate Eq. (5) 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,

$$\begin{aligned} t \cdot \mathbf{w}(m) = \quad & t \cdot \left\{\textstyle\sum_{m'>m} 1_{P(m,m')>0} + \sum_{m'<m} 1_{-P(m',m)>0}\right\} \\ & + s \cdot \left\{\textstyle\sum_{m'>m} 1_{P(m,m')=0} + \sum_{m'<m} 1_{P(m',m)=0}\right\}\,. \end{aligned} \qquad (6)$$

Eq. (6) 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. (6), as well as in the bits of equality to zero in the second sum, as described in Section 3.4. As the value of $t \cdot \mathbf{w}(m)$ is a linear combination of those bits, the talliers can then use the secret shares in those bits and Eq. (6) 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 3.3) 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 $\sum_{m=M-K}^{M-1} m = K \cdot \left(M - \frac{K+1}{2}\right)$. That 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 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 $1 \le m < m' \le M$. 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. (6); 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 winning candidates (Lines 8-10).

We note that if we set $K = M$ then Sub-protocol 2 outputs a full ranking of the M candidates. Alternatively, if the goal is to get the K winning candidates, for some predetermined $K < M$, without revealing their order, then the last stage in Sub-protocol 2 can be modified in order to meet that goal.

Sub-Protocol 2: Determining the winners in Copeland rule

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

1 **forall** $1 \le m < m' \le M$ **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 $w_d(m) = t \cdot \left\{\sum_{m'>m} \sigma_d^+(m,m') + \sum_{m'<m} \sigma_d^-(m',m)\right\} + s \cdot \left\{\sum_{m'>m} \sigma_d^0(m,m') + \sum_{m'<m} \sigma_d^0(m',m)\right\}$;
8 **forall** $k \in [K] := \{1,\ldots,K\}$ **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 kth elected candidate;
10 The talliers output the candidate that was found and remove him from $\mathbf{C}$;

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

4.5.1 Privacy. 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 $D' = \lfloor (D+1)/2 \rfloor$, 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 3.3–3.4). The secure comparison sub-protocol is perfectly secure, as was shown in [25]. 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 [14] and [11], that was shown there to be secure. Hence, Sub-protocol 2 is perfectly secure.

4.6 An MPC computation of the winners in the Minimax rule

Fixing $m \in [M]$, the talliers need first to find the index $m' \ne m$ 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 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 Minimax 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 4.5.1.

4.7 The protocol's security

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 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.

Another possible attack scenario is as follows: 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 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.

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. However, more independent and reputable talliers would be needed, and the communication and computational costs of our protocol would increase, albeit modestly.

5 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.

#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.

We set the number of talliers to be $D \in \{3, 5, 7, 9\}$. As explained in Section 4.7, larger values of D provide greater security, as well as higher runtimes. We set the number of candidates to be $M \in \{5, 10, 20\}$. The number N of voters 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 by N/B in order to get the overall time for validating all incoming ballots. As for the size p of the underlying field, we chose the prime $p = 2^{31} - 1$. It is sufficiently large for our purposes (see a detailed discussion in Appendix F).

5.1 The cost of batch validation of ballots

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. For more details see Appendix E.5.

Chida et al. [11] 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, $p = 2^{31} - 1$, are faster; but since we are interested only in the upper-bound of the computational overhead of our protocol, in order to establish its practicality, their chosen p is sufficient. 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 the case where the batch size is $B = 500$. To do that, it is necessary

	$B = 500$	$B = 5000$	$B = 25000$
$M = 5$	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	160/188/219/268	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 $D = 3/5/7/9$.

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.

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 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 runtimes for validating incoming ballots are realistic and are not expected to slow down the election process.

5.2 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 as well as $M(M-1)/2$ invocations of the equality to zero test (Section 3.4), followed by $K \cdot \left(M - \frac{K+1}{2}\right) \leq M(M-1)/2$ secure comparisons (Section 3.3). As discussed in Section 3.4, the cost of a secure comparison is the upper bound to the costs of the MPC computations to determine positivity and equality to zero. Hence, $4M(M-1)$ secure comparisons is the upper bound to the cost of Sub-protocol 2.

In order 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.

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 4.6). In summary, it is possible to achieve perfect ballot secrecy, as our protocol offers, at a very small computational price.

6 CONCLUSION

In this study we presented a protocol for the secure computation of order-based voting rules. Securing the voting process is an essential step towards a fully online voting process, which is needed more than ever in these current times of social distancing.

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 an 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 in order 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. Ours is the first study that offers a privacy-preserving solution for order-based voting rules. We specifically demonstrated our solution on two order-based rules: Copeland and Maximin. We showed that our solution is lightweight and can thus be readily implemented in real-life order-based elections.

The present study suggests several directions for future research:

(a) To demonstrate the features and advantages of our protocol, we intend to implement a running voting system, as was done in [16] for the secure protocols implementing score-based rules [15].

(b) Multi-winner election rules are designed specifically for selecting candidates that would satisfy the voters the most [17, 19], in the sense that they also comply with additional social conditions (e.g., that the selected winners include a minimal number of representatives of specific gender, race, region etc.). This problem has unique features and it therefore requires its own secure protocols.

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

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] J.C. Benaloh. 1986. Secret sharing homomorphisms: Keeping shares of a secret secret. In *CRYPTO*. 251–260.

[4] G.R. Blakley. 1979. Safeguarding Cryptographic Keys. In *International Workshop on Managing Requirements Knowledge*. 48: 313–317.

[5] 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.

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

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

[8] 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.

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

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

[11] 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.

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

[13] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. 2010. A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security* 9, 6 (2010), 371–385.

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

[15] 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.

[16] Lihi Dery, Tamir Tassa, Avishay Yanai, and Arthur Zamarin. 2021. A Secure Voting System for Score Based Elections. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2399–2401.

[17] 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*.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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).

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

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

A EMULATING MULTIPLICATION GATES

Here we describe how the generic secret-sharing-based protocol of Damgård and Nielsen [14] emulates multiplication gates. That computation requires the interacting parties to generate shares in a random field element r, such that r distributes uniformly on $\mathbf{Z}_p$ 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 ρ_d and performs a D'-out-of-D sharing of it among $T_1, \ldots, T_D$. At the completion of this stage, each T_d adds up all the D shares that he received and obtains a value that we denote by r_d. It is easy to see that $\{r_1, \ldots, r_D\}$ is a D'-out-of-D sharing of the random value $\rho = \sum_{d \in [D]} \rho_d$. Clearly, ρ 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 $\{u_1, \ldots, u_D\}$ be D'-out-of-D shares in u, which were generated by a polynomial $f(\cdot)$ of degree $D'-1$, and $\{v_1, \ldots, v_D\}$ 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, $\{w_1, \ldots, w_D\}$ 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 $\{r_1, \ldots, r_D\}$, and a $(2D'-1)$-out-of-D sharing, denoted $\{R_1, \ldots, R_D\}$. Next, each T_d computes $\tilde{w}_d = w_d + R_d$ and sends the result to T_1. Since $\{\tilde{w}_1, \ldots, \tilde{w}_d\}$ 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. □

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, $(-2x \mod p) = -2x$. As that number is even, its LSB is 0. □

D PROOF OF THEOREM 3

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 $Q = (Q(m, m'))_{1 \leq m, m' \leq M}$, 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 mth 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 mth 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 $\mathbf{k} := (k(1), \ldots, k(M))$ 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. □

E VERIFYING THE LEGALITY OF THE CAST BALLOTS

This section provides a complete discussion of the sub-protocol that the talliers may invoke in order to verify the legality of the cast ballots. A short summary was provided in the body of this manuscript in Section 4.4.

E.1 Motivation and overview

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, that voter manages 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 does no longer correspond 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 an unfortunate case, it is possible that the set of winners that P would dictate differs from that which P' dictates. 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 a proof of the voter's dishonesty. The ability of constructing such proofs, that could be used in legal actions against dishonest voters, might deter voters from attempting cheating.

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.

E.2 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 $\{0, 1\}$) if and only if $(x+1)\cdot(x-1) = 0$ (resp. $x\cdot(x-1) = 0$). Hence, the talliers input their shares in x to an arithmetic circuit that outputs the product $(x+1)\cdot(x-1)$ for Copeland or $x\cdot(x-1)$ for Maximin. If the output of such a circuit 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.

E.3 Verifying condition 4

First, we make the following observation. Let $x, y \in \mathbf{Z}_p$ 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 $a, b \in \mathbf{Z}_p$ 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

$$Q_m = \sum_{m' \in [M]} Q(m', m) = \sum_{m' < m} Q(m', m) - \sum_{m < m'} Q(m, m') \,. \quad (7)$$

Since the talliers hold shares in $Q(m', m)$ for all $1 \le m' < m \le M$, they can use Eq. (7) 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

$$Q_m = \sum_{m' < m} Q(m', m) - \sum_{m < m'} Q(m, m') + (M - m) \,. \quad (8)$$

Here too, the linearity of sharing and the relation in Eq. (8) 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

$$F(Q) := \prod_{1 \le m' < m \le M} (Q_m - Q_{m'}) \,. \quad (9)$$

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

E.4 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 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 on the content of the ballot.

The procedure for verifying condition 4 in Theorems 3 and 4 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 3, see D. Hence, as can be readily verified, if Q is a legal ballot then the value of $F(Q)$, Eq. (9), which the talliers compute in the validation procedure, equals

$$F(Q) = \pm 2^{\binom{M}{2}} \cdot \prod_{1 \le m' < m \le M} (m - m') \,, \quad (10)$$

where the sign of the product in Eq. (10) 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. Such a leakage of information is meaningless, but it can be eliminated by squaring the result and recovering $F(Q)^2$.

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, $(Q_1, \ldots, Q_M)$ 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 $(0, 1, \ldots, M-2, M-1)$.

E.5 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.

F A LOWER BOUND ON THE UNDERLYING 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:

- D, as that is the number of talliers (see Section 3.1).
- $2N$, 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 $[-N, N]$.
- $\max\{t, s\} \cdot (M-1)$, since that is the upper bound on $t \cdot \mathbf{w}(m)$, see Eq. (5), which is secret-shared among the talliers.
- $2(M-1)$, since in validating a given ballot matrix Q, the talliers need to test the equality to zero of $F(Q)$, see Eq. (9). 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. (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.