Fear Not, Vote Truthfully: Secure Multiparty Computation of Score Based Rules

Lihi Dery[a,b,*], Tamir Tassa[c], Avishay Yanai[d]

[a]*Department of Industrial Engineering and Management, Ariel University, Ariel, Israel*
[b]*Ariel Cyber Innovation Center, Ariel, Israel*
[c]*Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel*
[d]*VMware Research*

Abstract

We propose a secure voting protocol for score-based voting rules, where independent talliers perform the tallying procedure. The protocol outputs the winning candidate(s) while preserving the privacy of the voters and the secrecy of the ballots. It offers perfect secrecy, in the sense that apart from the desired output, all other information – the ballots, intermediate values, and the final scores received by each of the candidates – is not disclosed to any party, including the talliers. Such perfect secrecy may increase the voters' confidence and, consequently, encourage them to vote according to their true preferences. The protocol is extremely lightweight, and therefore it can be easily deployed in real life voting scenarios.

Keywords: Electronic Voting, Secure Multiparty Computation, Perfect Ballot Secrecy, Voting Protocols, Computational Social Choice

---

*Corresponding author
Email addresses: lihid@ariel.ac.il (Lihi Dery), tamirta@openu.ac.il (Tamir Tassa), yanaia@vmware.com (Avishay Yanai)

1. Introduction

Ballot secrecy is an essential goal in the design of voting systems. When voters are concerned for their privacy, they might decide to vote differently from their real preferences, or even abstain from voting altogether. Our main goal here is achieving perfect ballot secrecy. The usual meaning of privacy in the context of secure voting is that the voters remain anonymous. Namely, even though the ballots are known (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. We go one step further and consider 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.

The mere anonymity of the ballots might not provide sufficient privacy and hence may encourage untruthful voting, as our next two examples show. Consider a group of faculty members who need to jointly decide which applicant to accept to the faculty out of a given list of candidates. To that end, each faculty member (voter) anonymously casts a ballot. A tallier counts the ballots and uses some voting rule to determine the elected candidate(s). The problem with this voting strategy is that even though the tallier cannot link voters to ballots, he does see the actual ballots. Hence, besides the final outcome, say, that Alice is the elected candidate, the tallier is exposed to additional information which may be sensitive; e.g., that the candidate Bob received no votes, even though some of the voters declared upfront that they are going to vote for him. The imperfect privacy of such a voting system may cause some voters to vote untruthfully. The protocol that we present herein offers perfect privacy and, thus, may encourage voters to vote truthfully.

As another example, consider the London Inter-Bank Offered Rate (LIBOR)[1] which is the benchmark interest rate at which banks can borrow from each other. The rate is computed daily; banks that are benchmark submitters contribute to setting the LIBOR by means of voting: each bank's "vote" is an interest rate and the LIBOR is determined by some averaging over the submitted votes. The bank's submitted rate may signal the bank's financial viability. Worrying about the signal which their submitted rate conveys, some banks may submit an untruthful rate. To prevent this, the bank's individual submissions (the ballots) are kept private and are published only three months after the submission date. However, the tallier is exposed to these ballots and may be able to link some ballots to banks by financial

---

[1]See ICE LIBOR https://www.theice.com/iba/libor

analysis. Therefore, even anonymous ballots might not provide sufficient privacy. Securing the ballots, as we suggest herein, means that there is less incentive to misrepresent one's ballot, and thus there is less incentive for strategic voting.

Contributions. We present a secure protocol with perfect ballot secrecy to compute election results for score-based voting rules. This is achieved by employing cryptographic multiparty computation techniques. Score-based voting rules are rules where a voter's ballot consists of scores given to each of the candidates, and the winner is the candidate with the highest aggregated score (Brandt et al., 2016). This family includes rules such as PLURALITY, RANGE, APPROVAL, VETO, and BORDA. We follow what is known in cryptography as 'the mediated model" (Alwen et al., 2008), in the sense that our protocol involves a set of talliers who perform the aggregation of ballots and compute the final voting results, but they are not allowed to access the actual ballots or other computational results such as the final scores of candidates. Our protocol is secure against coalitions: in order to infer any information on aggregated scores of candidates, *at least half* of the talliers would need to collude, while in order to obtain the actual private ballots, *all* talliers would need to collude. Such perfect ballot privacy, by which the ballots and aggregated scores are not disclosed even to the talliers, may increase the voters' confidence and, consequently, encourage them to vote according to their true preferences. As the protocol is compliant with all desired properties of secure voting systems, and is very efficient, it can be readily implemented in real life voting scenarios.

The paper is organized as follows. In Section 2 we review related work. In Section 3 we provide the necessary preliminaries on score-based voting rules and on secret sharing schemes. Our protocol is presented and discussed in Section 4. We analyze the computational and communication costs of the protocol in Section 5, discuss the protocol's compliance with essential electronic voting requirements in Section 6, and conclude in Section 7.

2. Related work

The issue of enhancing democratic elections is widely studied in the AI community and specifically in the computational social choice community. Some recent studies look at securing attacks during the recounting of ballots (Elkind et al., 2019), optimal attack problems in voting (e.g. by deleting voters (Dey et al., 2019)), electoral bribery problem (Chen et al., 2019), election control through social influence (Corò et al., 2019) and the complexity of multi-winner voting rules (Yang, 2019). A central goal in this context is the design of systems in which

the election results reflect properly and truthfully the will of the individuals in the underlying society. An important vehicle towards achieving that goal is to secure the voting system so that it provides desired properties such as anonymity/privacy, fairness, robustness, uniqueness, and uncoercibility.

Previous studies on secure voting focused on different desired properties, e.g. privacy, or anonymity (a ballot cannot be connected to the voter who cast it), uniqueness (every voter can vote once), correctness (the issued winners are the ones that should be selected by the underlying voting rule from the cast ballots), and fairness (all voters must cast their ballot without seeing other votes or intermediate voting results). See e.g. Chang & Lee (2006); Gritzalis (2002); Zagórski et al. (2013).

Our focus is on preserving privacy and achieving perfect ballot secrecy. One way to achieve those goals is by using methods that allow the voters to compute the outcome themselves without relying on a tallier to aggregate and count the votes, e.g. (Benkaouz & Erradi, 2015). Another way is to use a third-party, a.k.a *a tallier*. In order to secure the transition of the votes which are sent from the voters to the tallier, various cryptographic techniques were utilized in prior art.

Early studies used the notions of mix-nets and anonymous channels (Chaum, 1988; Park et al., 1993; Sako & Kilian, 1995). Blind signatures (Chaum, 1982) were used in other secure e-voting protocols, e.g. (Fujioka et al., 1992; Ibrahim et al., 2003). Chen et al. (2014) proposed a secure e-voting system based on the hardness of the discrete logarithm problem. Benaloh (1987) proposed a practical scheme for conducting secret-ballot elections in which the outcome of an election is verifiable by all participants and even by non-participating observers; his scheme is based on secret sharing homomorphisms (Benaloh, 1986) that allow computations on shared data.

A large number of studies utilized homomorphic encryption, as it enables voting aggregation in the ciphertext domain. For example, Cramer et al. (1997) proposed a scheme in which each voter posts a single encrypted ballot; owing to the homomorphism of the cipher, the final tally is verifiable to any observer of the election. Damgård et al. (2010) proposed a generalization of Paillier's probabilistic public-key system (Paillier, 1999) and then showed how it can be used for efficient e-voting. While most homomorphic e-voting schemes are based on additive homomorphism, Peng et al. (2004) proposed a scheme based on multiplicative homomorphism. In their scheme, the tallier recovers the product of the votes, instead of their sum, and then the product is factorized to recover the votes.

To the best of our knowledge, only two previous studies considered the question of private execution of the computation that the underlying voting rule dic-

tates. Canard et al. (2018) considered the Majority Judgment (MJ) voting rule (Balinski & Laraki, 2007), which does not fall under the score-based family of rules that we consider here. They first translate the complex control flow and branching instructions that the MJ rule entails into a branchless algorithm; then they devise a privacy-preserving implementation of it using homomorphic encryption, distributed decryption schemes, distributed evaluation of Boolean gates, and distributed comparisons. Nair et al. (2015) suggest 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. In our study, which covers all score-based rules, we provide perfect privacy as well as means for preventing cheating by using a secret sharing-based secure multiparty computation (see Section 4.1).

3. Preliminaries

This section provides the required background on score based voting rules (Section 3.1) and secret sharing (Section 3.2).

3.1. Score-based voting rules

We consider a setting in which there are N voters, $\mathbf{V} = \{V_1, \ldots, V_N\}$, that need to hold an election over M candidates, $\mathbf{C} = \{C_1, \ldots, C_M\}$. The election determines a score $\mathbf{w}(m)$ for each candidate C_m, $m \in [M] := \{1, \ldots, M\}$ in a manner that will be discussed below. Let $K \in [M]$ be some fixed integral parameter. Then the output of the voting algorithm is the subset of the K candidates with the highest $\mathbf{w}$-scores, where ties are broken either arbitrarily or by another rule that is agreed upfront. ($K = 1$ corresponds to the typical case of a single winner.) Our protocol can be easily extended to output also the ranking of the candidates or the final scores they received. As such extensions are straightforward, we focus here on the "lean" output consisting only of the identities of the K elected candidates.

In score-based voting rules, every voter V_n, $n \in [N] := \{1, \ldots, N\}$, creates a ballot vector of the form $\mathbf{w}_n := (\mathbf{w}_n(1), \ldots, \mathbf{w}_n(M))$, where all single votes, $\mathbf{w}_n(m)$, are nonnegative and uniformly bounded. Define

$$\mathbf{w} = (\mathbf{w}(1), \ldots, \mathbf{w}(M)) := \sum_{n=1}^{N} \mathbf{w}_n \,. \tag{1}$$

Then $\mathbf{w}(m)$ is the aggregated score of the candidate C_m, $m \in [M]$.

We consider five types of voter inputs to be used in the above described rule template, which give rise to five well known voting rules (Nurmi, 2012):

- PLURALITY. $\mathbf{w}_n \in \{\mathbf{e}_1, \ldots, \mathbf{e}_M\}$ where $\mathbf{e}_m$, $m \in [M]$, is an M-dimensional binary vector of which the mth entry equals 1 and all other entries are 0. Namely, V_n casts a vote of 1 for exactly one candidate and a vote of 0 for all others, and the winner is the candidate who was the favorite of the maximal number of voters.

- RANGE. $\mathbf{w}_n \in \{0, 1, \ldots, L\}^M$ for some publicly known L. [2] Here every voter gets to give a score, ranging from 0 to L, to each candidate.

- APPROVAL. $\mathbf{w}_n \in \{0, 1\}^M$. Every voter submits a binary vector in which (up to) K entries are 1, while the remaining entries are 0. Such a voting rule is used when it is needed to fill K equivalent positions; for example, if K members in the senate of a university retired, it is needed to select K new senate representatives from the faculty.

- VETO. $\mathbf{w}_n \in \{\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_M\}$ where $\hat{\mathbf{e}}_m$, $m \in [M]$, is an M-dimensional binary vector of which the mth entry equals 0 and all other entries are 1. In this method every voter states his least preferred candidate. The winner is the candidate that got the minimal number of zero votes.

- BORDA. $\mathbf{w}_n \in \{(\pi(0), \ldots, \pi(M-1)) : \pi \in \Pi_M\}$, where Π_M is the set of all permutations over the set $\{0, \ldots, M-1\}$. Here, the input of each voter is his own ordering of the candidates, i.e., $\mathbf{w}_n(m)$ indicates the position of C_m in V_n's order, where a position of 0 (resp. $M-1$) is reserved to V_n's least (resp. most) favorite candidate.

3.2. Secret sharing

Secret sharing methods (Shamir, 1979) 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 (1979) and Blakley (1979), for the case of threshold secret sharing. Assuming that there

---

[2]We note that RANGE is not commonly included in the family of score-based voting rules, but we include it in this family since it fits the same voting rule "template". RANGE is common in many applications, e.g. www.netflix.com and www.amazon.com, and it is often used in recommender systems (Masthoff, 2011).

are D participants, $\mathbf{P} = \{P_1, \ldots, P_D\}$, then the access structure in Shamir's and in Blakley's schemes consists of all subsets of $\mathbf{P}$ of size at least D', for some $D' \leq D$. Such secret sharing schemes are called D'-out-of-D.

Our protocol utilizes a simpler all-or-nothing secret sharing scheme. That scheme is a D-out-of-D one, in the sense that it allows a recovery of the secret only if *all* participants (who in our case are the different talliers) combine their shares. Any other subset of shares reveals no information at all on the secret.

Assuming that the secret w to be shared is a bounded integer, we can view it as an element in a sufficiently large finite field $\mathbf{Z}_p$. In order to split the secret w among a group of D participants, the owner of the secret selects, independently and uniformly at random, $D - 1$ elements from $\mathbf{Z}_p$, denoted $w_1, \ldots, w_{D-1}$. He then sets $w_D := w - \sum_{d=1}^{D-1} w_d \mod p$. The value w_d is the share given to the dth participant, $d \in [D]$. It is easy to see that the combination of all shares enables to recover the secret w, since $w = \sum_{d=1}^{D} w_d \mod p$. However, any tuple of D' shares, where $D' < D$, reveals zero information on w, since such tuples of shares distribute uniformly at random in $\mathbf{Z}_p^{D'}$.

Our protocol involves a *distributed* third party, $\mathbf{T} = \{T_1, \ldots, T_D\}$, called the tallier ($\mathbf{T}$) or talliers ($T_d$, $d \in [D] = \{1, \ldots, D\}$). In the protocol, we use secret sharing for creating shares of the private ballots of the voters and distributing them among the D talliers. As those ballots are vectors (see Section 3.1), the secret sharing is carried out for each entry independently, so that each of the talliers receives a (vector) share in each ballot.

We note that our protocol may be easily modified to work with Shamir's threshold secret sharing scheme (Shamir, 1979), instead of the above described all-or-nothing scheme. In that variant, the shares are computed so that any number of $t < D$ talliers, for some predetermined threshold t, can perform the tallying computation, in order to enhance the robustness (also known as 'availability') of the system.

4. The method: A secure protocol for score based rules

In this section we now present our protocol. As indicated earlier, our protocol is mediated, in the sense that it assumes a set of talliers, $\mathbf{T} = \{T_1, \ldots, T_D\}$, who assist in the computations, but are not allowed to learn any information on the private votes of the voters. The number of talliers, D, can be any integer $D > 1$. Higher values of D will imply higher computational and communication costs, but they will also imply greater security against coalitions of corrupted talliers.

A privacy-preserving implementation of score-based rules is described in Protocol 1. Before delving into that protocol, we make the following observation. For each of the five score-based rules, there is a known upper bound B on the entries of $\mathbf{w}$. $B = N$ in PLURALITY, APPROVAL, and VETO rules, $B = NM$ in BORDA, and $B = NL$ in RANGE. Let p be a fixed prime greater than B. Then all computations in Protocol 1 are carried out in the field $\mathbf{Z}_p$.

First, each voter V_n, $n \in [N]$, constructs his own ballot vector (Step 1), $\mathbf{w}_n \in (\mathbf{Z}_p)^M$. We assume that all voters know the index $m \in [M]$ of each candidate. For example, that index can be determined by the lexicographical ordering of the candidates according to their names. In Step 2, V_n creates random additive shares of $\mathbf{w}_n$. Specifically, V_n selects uniformly at random D vectors in $(\mathbf{Z}_p)^M$ such that their sum equals $\mathbf{w}_n$. Letting $\mathbf{w}_{n,d}$ be the dth share of $\mathbf{w}_n$, then

$$\mathbf{w}_n = \sum_{d=1}^{D} \mathbf{w}_{n,d} \mod p\,. \tag{2}$$

Subsequently, each V_n sends the dth share to T_d, $d \in [D]$ (Step 3). After receiving the ballot shares from all voters, T_d computes the sum of all received vector shares $\hat{\mathbf{w}}_d = \sum_{n=1}^{N} \mathbf{w}_{n,d} \mod p$ (Step 4). Each such vector $\hat{\mathbf{w}}_d$ on its own carries no information regarding the votes (since it is the sum of uniformly random and independent vectors). But in view of Eq. (1) and the way in which the share vectors $\mathbf{w}_{n,d}$ were generated (Step 2), we see that

$$\mathbf{w} = \sum_{d=1}^{D} \hat{\mathbf{w}}_d \mod p\,, \tag{3}$$

where $\mathbf{w}$ is the aggregated vector of scores.

The heart of the protocol is in Step 5: here, the talliers engage in a secure multiparty computation (MPC) in order to find the indices of the K candidates with the highest aggregated scores. This is a non-trivial task since no one holds the vector $\mathbf{w}$. In Section 4.1 we explain the notion of MPC and describe the MPC protocol that we use for the above described task. Once those indices are found, the voters proceed to find the identity of the K candidates behind those indices (Step 6).

4.1. Sorting shared vectors

The main challenge in Step 5 of Protocol 1 is to find the indices of the K largest entries in $\mathbf{w}$ without revealing any information on $\mathbf{w}$'s entries. Towards

8

Protocol 1 A basic protocol for secure score-based voting

Input: $\mathbf{w}_n$, $n \in [N]$; $K \in [M]$.

Output: The K candidates from $\mathbf{C}$ with highest aggregated scores in $\mathbf{w} = \sum_{n=1}^{N} \mathbf{w}_n$.

1: Each voter V_n, $n \in [N]$, constructs his ballot vector $\mathbf{w}_n$ according to the selected indexing and voting rule.
2: Each voter V_n, $n \in [N]$, selects uniformly at random $D - 1$ vectors $\mathbf{w}_{n,d} \in (\mathbf{Z}_p)^M$, $1 \le d \le D - 1$, and then sets $\mathbf{w}_{n,D} = \left(\mathbf{w}_n - \sum_{d=1}^{D-1} \mathbf{w}_{n,d} \right) \mod p$.
3: V_n, $\forall n \in [N]$, sends $\mathbf{w}_{n,d}$ to T_d, $\forall d \in [D]$.
4: T_d, $\forall d \in [D]$, computes $\hat{\mathbf{w}}_d = \sum_{n=1}^{N} \mathbf{w}_{n,d} \mod p$.
5: $T_1, \ldots, T_D$ find the indices of the K candidates in $\mathbf{C}$ with highest $\mathbf{w}$-scores and output them.
6: The voters find the identities of the top K candidates.

---

that end, the talliers can implement any sorting algorithm on $\mathbf{w}$ until all K largest entries are found. To do so, the talliers need to compare entries in $\mathbf{w}$; alas, those entries are unknown to them, as the talliers hold only random shares in them. This raises the question of how such comparisons can be carried out without revealing to the talliers the actual values that they obliviously compare.

Assume that the two entries that need to be compared are $u = \mathbf{w}(n)$ and $v = \mathbf{w}(m)$ for some $n, m \in [M]$. Each tallier T_d, $d \in [D]$, holds random shares $u_d, v_d \in \mathbf{Z}_p$, such that $u = \sum_{d=1}^{D} u_d \mod p$ and $v = \sum_{d=1}^{D} v_d \mod p$. The talliers wish to find whether $v < u$, but without revealing any information beyond that on u and v.

To privately verify such questions, we use a secure multiparty computation (MPC) protocol (Yao, 1982). An MPC protocol allows $T_1, \ldots, T_D$ to compute any function f over private inputs $x_1, \ldots, x_D$ that they hold, so that at the end of the protocol everyone learns $f(x_1, \ldots, x_D)$ *but nothing else*.[3] A common approach towards designing efficient MPC protocols is to represent the function f by an arithmetic circuit C such that for every set of inputs, $x_1, \ldots, x_D$, the output of the

---

[3]Of course, some information may be inferred from the desired output $f(x_1, \ldots, x_D)$, but this is inevitable and allowed. For example, if the desired output is a median of $x_1, \ldots, x_D$, then at the completion of the protocol, every tallier whose input is smaller than the median can infer that there are at least $\frac{D}{2}$ talliers that hold greater values than his own.

circuit, $C(x_1, \ldots, x_D)$, equals $f(x_1, \ldots, x_D)$.

The circuit is composed of input and output gates such that each input gate is fed with a single secret value by one of the parties, and an output gate determines a single value that is revealed to all parties (there could be multiple output gates). Additionally, between the input and output gates there are multiple layers of arithmetic gates that connect them. An arithmetic gate can be either addition or multiplication. Each gate is given exactly two inputs, and it produces one output such that the output of a gate at layer ℓ can be given as input to multiple gates in layer $\ell+1$. (All input gates constitute the first layer of the circuit.) Only the values that the output gates issue are revealed to the parties; all intermediate values that pass from one gate to another remain secret from everyone. Specifically, a secure protocol allows the parties to maintain the invariant that the actual value output from each gate is secret-shared, as described in Section 3.2. When reaching an output gate, each party broadcasts the corresponding share that it holds, so that everyone can reconstruct the output.

The computational and communication costs of computing such circuits depend mainly on the number of multiplication gates and on the number of layers in the circuit, as we proceed to explain. To compute a multiplication gate of two secrets, s_1 and s_2, the parties have to interact; i.e., each party needs to send some information to the other parties. However, to compute a multiplication gate of one secret s and a public value c, the parties do not need to interact (such a gate requires only local computation). The same holds for an addition gate. Therefore, for the efficiency of secure computation, circuit designers are mostly concerned with the number of multiplication gates in the circuit and with the *depth* of the circuit, i.e. the number of interactions that have to be performed sequentially (since they depend on each other and cannot be performed in parallel). Specifically, in this work we use the design by Nishide & Ohta (2007), which outputs the bit that indicates whether $u < v$ by a circuit with a *constant depth* (15 to be concrete). This is advantageous as the depth does not depend neither on the number of parties nor on the field size p, so changing those parameters does not have a significant effect on performance (see Table 1 in Section 5).

4.2. The protocol's security

Here we discuss the security of the whole protocol. An important goal of secure voting is to provide anonymity; namely, it should be impossible to connect a ballot to the voter who cast it. Protocol 1 achieves that goal since each cast ballot is distributed into random shares and then each share is sent to a different tallier. Each such share carries zero information on the underlying ballot. Even subsets

of d shares where $d < D$ reveal no information on the secret ballot. Hence, the talliers never obtain access to the actual ballots and, consequently, cannot infer anything about the way each voter voted. The only way in which the ballot of a voter V_n can be recovered is if *all* talliers collude and recover his ballot $\mathbf{w}_n$ by adding all of the corresponding shares, see Eq. (2).

Such a collusion scenario 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 $D = 4$ or even $D = 3$, the probability of a total breakdown of trust in any conceivable application scenario (with a proper selection of the talliers) would be negligible.

Another possible attack scenario is as follows: a voter V_j can eavesdrop on the communication link between another voter V_n and each of the talliers, and intercept the messages that V_n sends to the talliers (in Protocol 1's Step 3) in order to recover $\mathbf{w}_n$ from them; additionally, V_j may replace V_n's original messages that carry shares of $\mathbf{w}_n$ with other messages (say, ones that carry shares of $\mathbf{w}_j$, or any other desired fake ballot). Such an attack 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 Step 3 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.

Finally, the security of the computation that the talliers carry out in Step 5 of Protocol 1 is ensured by the proven security of the underlying secure multiparty computation. By utilizing that protocol, the talliers may find the indices of the K winning candidates without learning any information beyond the order that the aggregated scores induce on the candidates[4]. The security of that computation is guaranteed under the assumption of an honest majority, namely, that if some of the talliers collude, the number of colluding talliers is smaller than $D/2$. So, for example, our voting protocol with $D = 3$ (resp. $D = 5$) talliers is secure unless

---

[4]The circuit that we use in order to verify inequalities may be modified in order to hide intermediate comparison results and output only the K indices of candidates with highest scores. Such a version, which we do not describe herein, will output only the K winning candidates without disclosing their order.

at least two (resp. three) of them collude. It is important to note that in case there is a collusion between $D/2$ or more talliers, they could obtain information on aggregated scores of candidates. However, in order to get information on private ballots of specific voters, *all* talliers have to collude, since those ballots are shared through an all-or-nothing secret sharing scheme (Steps 2-3 of Protocol 1).

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, increasing D has its costs: more independent and reputable talliers are needed, and the communication and computational costs of our protocol increase, albeit modestly (see Section 5).

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. 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 would be disclosed to anyone.

4.3. Validating the legality of the cast ballots

Protocol 1 is designed for honest voters, namely, voters who cast legal votes. However, voters may attempt cheating by submitting illegal ballots in order to help their candidate of choice. For example, assume that V_n's favorite candidate is C_m and the voting rule is PLURALITY. Then a honest V_n would cast the ballot $\mathbf{w}_n = \mathbf{e}_m$ (see Section 3.1). A dishonest V_n, on the other hand, could cast the ballot $\mathbf{w}_n = N\mathbf{e}_m$. Such an illegal ballot would boost C_m's chances of winning, or, if V_n is the only dishonest voter, it would even ensure C_m's win. Similar options of cheating exist also with the other voting rules. Since the talliers do not see the actual ballots, if a voter can pull such a cheat, it might remain undetected.

In real-world voting scenarios, 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, we proceed to describe an MPC solution that enables the talliers to validate the legality of each ballot, even though those ballots remain hidden from them.

In case a ballot is found to be illegal, the talliers may recover it (by adding up all shares) and use the recovered ballot as a proof of the voter's dishonesty.

Let us start by examining the PLURALITY rule. A ballot $\mathbf{w}_n$ is legal in this case iff $\mathbf{w}_n(m) \leq 1$, i.e. if $\mathbf{w}_n(m) \in \{0, 1\}$) for all $m \in [M]$, and $\sum_{m \in [M]} \mathbf{w}_n(m) = 1$ mod p (assuming $p > M$). Each of the above m inequalities can be verified by a circuit that outputs the product $\mathbf{w}_n(m) \cdot (\mathbf{w}_n(m) - 1)$. The talliers accept the vote $\mathbf{w}_n(m)$ as legal (namely, being either 0 or 1) iff the result is 0. Finally, verifying that there exists exactly one entry in the vector $\mathbf{w}_n$ with the value 1 is done by computing the sum $\sum_{m \in [M]} \mathbf{w}_n(m)$ and verifying that it equals 1. The fact that $N < p$ ensures that there will not be a wrap around.

To validate ballots $\mathbf{w}_n$ in the case of VETO or APPROVAL, we also need to check that each entry, $\mathbf{w}_n(m)$, $m \in [M]$, is either 0 or 1, as described above for PLURALITY. If all entries were validated, the talliers need to proceed and check an aggregated condition on the ballot's entries. The aggregated condition in VETO can be checked by computing the sum $\sum_{m \in [M]} \mathbf{w}_n(m)$ and verifying that it equals $M - 1$. The aggregated condition in APPROVAL requires that the sum $S_n := \sum_{m \in [M]} \mathbf{w}_n(m)$ is at most K. That condition can be checked by a circuit that outputs $S_n \cdot (S_n - 1) \cdots (S_n - K)$. The talliers will accept $\mathbf{w}_n$ as legal iff the latter product equals 0. Indeed, if that product equals 0 then the talliers can deduce that V_n had voted for at most K candidates, without knowing the exact number of candidates for whom V_n had voted. Any result other than 0 will testify that V_n had cheated; in such cases the talliers can reject his vote and consider further consequences.

In the case of RANGE, a ballot $\mathbf{w}_n$ is legal iff $\mathbf{w}_n(m) \leq L$ for all $m \in [M]$. Each of the these m inequalities can be verified by a circuit that outputs $\mathbf{w}_n(m) \cdot (\mathbf{w}_n(m) - 1) \cdots (\mathbf{w}_n(m) - L)$. The talliers will accept the ballot $\mathbf{w}_n$ iff the result of that circuit will be 0 for each $m \in [M]$.

The BORDA rule is slightly different. A ballot is legal under this rule if it consists of some permutation of the M values in $\{0, 1, \ldots, M - 1\}$. Hence, a ballot is legal iff it satisfies the following two conditions:

$$\mathbf{w}_n(m) \leq M - 1 \quad \forall m \in [M]; \text{ and} \tag{4}$$

$$\mathbf{w}_n(m) \neq \mathbf{w}_n(m') \quad \forall m > m' \in [M]. \tag{5}$$

The entry-wise conditions in Eq. (4) can be verified as described earlier. The global condition in Eq. (5) can be verified by verifying that each of the $\binom{M}{2}$ differences $\mathbf{w}_n(m) - \mathbf{w}_n(m')$, $m > m' \in [M]$, is nonzero. However, for privacy

reasons, the talliers must not recover those differences, since they would reveal the entire ballot. To prevent such leakage of information and still allow the verification, the talliers can generate $\binom{M}{2}$ random secret elements in the field $\mathbf{Z}_p$, denoted by $\rho_{m.m'}$, $m > m' \in [M]$, and then compute $\zeta_{m,m'} := \rho_{m,m'} \cdot (\mathbf{w}_n(m) - \mathbf{w}_n(m'))$. If the underlying field $\mathbf{Z}_p$ is large, then with high probability (of $1/p$) the selected $\rho_{m.m'}$ is nonzero. In such cases, $\zeta_{m,m'} \neq 0$ iff $\mathbf{w}_n(m) - \mathbf{w}_n(m') \neq 0$, and $\zeta_{m,m'}$ reveals no information at all on $\mathbf{w}_n(m) - \mathbf{w}_n(m')$ (since $\rho_{m,m'}$ can be any nonzero element in the underlying field).

Generating shares of a random secret multiplier is a simple task. In fact, such a protocol is executed by the talliers anyway as part of the secure computation protocol of Chida et al. (2018). Specifically, for securely computing a multiplication gate in that protocol, the talliers generate two random secret values in the field. Therefore, in our computational cost analysis (Section 5), we upper bound the cost of generating shares in a random secret by the cost of evaluating a multiplication gate (where in fact the latter cost is strictly higher than the former).

Lastly, we consider the case of false negatives. In the field $\mathbf{Z}_p$ there is a probability of $1/p$ that the resulting random element would be zero. In such a case, $\zeta_{m,m'}$ would be zero even though $\mathbf{w}_n(m) - \mathbf{w}_n(m') \neq 0$. Hence, the talliers would get a false alarm regarding a ballot of some voter, as if it contains two equal entries, when in fact all of the ballot's entries are distinct, as required. But the probability of such a false alarm is $1/p$. In Section 5 we present runtimes for the field $\mathbf{Z}_p$ with $p = 2^{61} - 1$; in such large fields the probability $1/p$ is negligible. Furthermore, even if for some difference $\mathbf{w}_n(m) - \mathbf{w}_n(m')$ the talliers get an indication that it equals zero, they can repeat the test with another random and independent multiplier $\rho_{m.m'}$. If even that additional test yields $\zeta_{m,m'} = 0$, the talliers can withold that ballot until its validity is verified (say, by performing additional independent tests until the probability of an error reduces to below some given threshold, or by revealing the value of the difference $\mathbf{w}_n(m) - \mathbf{w}_n(m')$).

5. Evaluation: Computational and communication costs

We analyze herein the computational and communication costs for the voters (Section 5.1), and for the talliers, where the latter discussion is separated to two parts — the cost for computing the final election results (Section 5.2) and the cost for validating the legality of the cast ballots (Section 5.3).

5.1. Costs for the voters

The costs for each voter are negligible, as a voter needs only to generate $M(D-1)\log_2 p$ random bits, perform $M(D-1)$ additions in $\mathbf{Z}_p$, and then send D messages of $M\log_2 p$ bits each (Protocol 1, Steps 2-3).

5.2. The talliers: the cost of computing the final election results

The cost in Step 4 of Protocol 1 is negligible ($(N-1)M$ additions in $\mathbf{Z}_p$), but the determination of the winners (Step 5) is more costly as it invokes a protocol for secure comparison. The number of multiplication gates in the comparison circuit is $279 \cdot \log p + 5$ in a circuit of depth 15. A secure evaluation of a multiplication gate incurs a communication of $12 \log p$ bits per tallier (according to Chida et al. (2018, Table 2)). Hence, the overall communication per tallier for a single comparison is roughly $3348 \cdot \log^2 p$ bits (or < 1.5 megabytes when $p = 2^{61} - 1$).

In order to evaluate the runtime of performing such a comparison, 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. As for the bound B on aggregated scores, which affects the runtime, we examined two cases: $B < p_1$ and $B < p_2$, where $p_1 := 2^{13} - 1$ and $p_2 := 2^{31} - 1$ are two Mersenne primes. Namely, in cases where there are few voters, or ballots' entries are small, and as a result $B < p_1$, we used the bound p_1; otherwise, we used the bound p_2, which seems to suffice for all conceivable application scenarios. Using Mersenne primes is advantageous in the context of secure computation, since multiplication of two field elements can be done without performing an expensive division. The results are presented in the first two rows of Table 1.

Note that those runtimes are for a single comparison. In order to determine the identity of the K winners, it is necessary to perform up to KM comparisons. We can see that even in large election scenarios that require choosing an underlying field of size p_2, with $D = 9$ talliers (for enhanced security), with large numbers of candidates, M, and selected winners, K, the election results can be determined within only few seconds.

5.3. The talliers: the cost of validating ballots

For ballot validation, we consider three rules — PLURALITY, RANGE, and BORDA. (The validation of ballots in the two remaining rules, VETO and AP-PROVAL, is similar to that in PLURALITY.)

We extrapolated the runtime for executing the validation procedure for the various rules from the experimental results reported by Chida et al. (2018). They experimented on a similar network setting, but used a larger field with $p = 2^{61} - 1$.

	$D = 3$	$D = 5$	$D = 7$	$D = 9$
$B < p_1 = 2^{13} - 1$	2.83	4.3	6.6	12.81
$B < p_2 = 2^{31} - 1$	9.07	9.54	9.64	15.0
Validating $5 \cdot 10^4$ PLURALITY ballot entries	41.3	42.2	52.9	65.55
Validating $5 \cdot 10^4$ RANGE ballot entries with $L = 20$	826	844	1058	1311
Validating $5 \cdot 10^4$ RANGE ballot entries with $L = 100$	4210	4945	5770	7050
Validating $5 \cdot 10^4$ BORDA ballots with $M = 5$	1652	1688	2116	2622
Validating $5 \cdot 10^4$ BORDA ballots with $M = 10$	7434	7596	9522	11799

Table 1: Rows 1-2 show runtimes (milliseconds) for a secure comparison protocol with a varying number of talliers, D, and two field sizes, p_1 and p_2. Rows 3-7 show runtimes for a batch validation of $5 \cdot 10^4$ ballot entries or full ballots for various voting rules.

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 parties (talliers in our case), are shown in Table 2.

#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 2: 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.

5.3.1. The PLURALITY rule

As described in Section 4.3, a PLURALITY ballot validation depends on a circuit that checks whether $\mathbf{w}_n(m) \leq 1$. As explained there, such a circuit requires only one multiplication gate and has depth 1. In addition, to verify that there is only one ballot entry with the value 1, we need only to perform summation, which requires no multiplication gates. Overall, a validation circuit of q ballots requires exactly qM multiplication gates in *one layer*. As explained earlier, the runtimes in row 1 of Table 2 are for a circuit with 10^6 multiplication gates that were spread evenly over 20 layers. Hence, the runtime of executing $10^6/20 = 5 \cdot 10^4$ multiplication gates in a single layer is obtained by dividing those runtimes by 20, as

shown in row 3 of Table 1. Those are the runtimes for Validating $5 \cdot 10^4$ PLURAL-ITY ballot entries (from several different voters). In fact, those runtimes constitute an upper bound on the actual runtimes for our application, since they were obtained with a field size of $p = 2^{61} - 1$ that is larger than p_1 or p_2 that suffice for our needs. Those numbers indicate that validation is an extremely lightweight task in the case of the PLURALITY rule (as well as VETO and APPROVAL). Validating 50 million ballot entries can be done in roughly one minute.

5.3.2. The RANGE rule

Here we evaluate the cost of validating a ballot in the RANGE rule, as a function of the maximum score, L, that a voter can give to a candidate. The validation of an entry in a RANGE ballot requires a circuit with L multiplications in a row (when implemented naïvely, without optimizations). Thus, the experiment in Chida et al. (2018) with a circuit that consists of a million multiplication gates spread over 20 layers (the runtimes of which are reported in row 1 of Table 2) captures exactly $5 \cdot 10^4$ RANGE ballots entries, in the case $L = 20$. We report those numbers in row 4 of Table 1. When L equals 100, we rely on the runtimes that are reported in Chida et al. (2018) for a circuit of a million multiplication gates spread over 100 layers, as appear in row 2 of Table 2. In terms of validating RANGE ballot entries, those are the runtimes for validating 10000 RANGE ballot entries with $L = 100$. We multiply those runtimes by 5, in order to get the same batch size as in the case of $L = 20$, and report the resulting runtimes in row 5 row of Table 1.

We infer that even though validating RANGE ballot entries, especially for large values of L, is more costly than validating PLURALITY ballot entries, the validation task remains a very practical one. For example, validating 50 million RANGE ballot entries, even for $L = 100$ and $D = 9$, can be done in under two hours. (Recall that the election process usually spans a long period, say one day. Hence, the validation process can be spread along the entire election period, by validating each time the batch of ballots that were cast since the last validation.)

5.3.3. The BORDA rule

Finally, we turn to evaluate the cost of validating a ballot in the BORDA rule, as a function of the number of candidates, M. As described in Section 4.3, validating a BORDA ballot consists of two stages. In the first stage we check that each ballot entry is in the range $[0, M - 1]$. This can be done by a circuit of depth $M - 1$ with an overall number of $M(M - 1)$ multiplication gates. In the second stage we check that the ballot is a permutation of $\{0, \ldots, M - 1\}$. This verification

consists of generating $\binom{M}{2}$ secret random sharings and a secure computation of $\binom{M}{2}$ multiplication gates (see Section 4.3) in a circuit with a single layer. The computation and communication required for generating shares of a secret random value are strictly smaller than those required for a multiplication gate, since in the protococl of Chida et al. (2018), evaluating a multiplication gate involves the generation of two secret random sharings. Thus, the cost of the second stage in the validation of a BORDA ballot can be bounded by the cost of performing $2 \cdot \binom{M}{2} = M(M-1)$ multiplications, in two consecutive layers. In total, the overall cost of both stages in validating a single BORDA ballot can be bounded by performing $2M(M-1)$ multiplications over $M-1$ layers.[5]

When $M = 5$, the overall cost of validating a single ballot can be bounded, as stated above, by performing $2M(M-1) = 40$ multiplications spread over 4 layers. Hence, to validate a batch of $2.5 \cdot 10^4$ BORDA ballots in this case we need to perform one million multiplications spread over 4 layers. Those runtimes, for the various values of D, can be bounded by the runtimes in row 1 of Table 2, since the latter runtimes are for the same number of multiplications, only spread over more layers (20 instead of 4). Finally, we conclude that the overall runtime for validating a batch of $5 \cdot 10^4$ BORDA ballots is given by doubling the runtimes in row 1 of Table 2, as shown in row 6 of Table 1.

Bounding the runtimes for $M = 10$ is done in a similar fashion. The overall cost of validating a single ballot is smaller than that of performing $2M(M-1) = 180$ multiplications spread over 9 layers. Hence, to validate a batch of $10^6/180$ BORDA ballots in this case we need to perform one million multiplications spread over 9 layers. Those runtimes, for the various values of D, can be bounded by the runtimes in row 1 of Table 2. Finally, we conclude that the overall runtime for validating a batch of $5 \cdot 10^4$ BORDA ballots is given by multiplying the runtimes in row 1 of Table 2 by 9, as shown in row 7 of Table 1.

In cases where $M \geq 22$, we can use the runtimes reported in rows 2 or 3 of Table 2, since in such cases the number of layers (in the first stage), which is $M-1$, would be greater than 20. However, as the BORDA rule is impractical for such values of M, because it is hard to expect voters to have a full ranking over such a large number of candidates, we leave it for the interested reader to

---

[5]Note that by taking a spread of the needed workload over more layers than what is actually needed, one gets higher runtimes. We perform that relaxation for the sake of simplicity, since we are only interested in upper bounds on the validation cost that will demonstrate its lightweight nature.

compute the resulting runtime bounds for such values of M. But the numbers for $M = 5$ and $M = 10$ indicate that the validation of BORDA ballots is a very practical task, despite the higher complexity of that task in comparison to that in PLURALITY or RANGE. For example, the validation of one million ballots when there are $M = 10$ candidates and $D = 9$ talliers would take less than 4 minutes.

6. Discussion

Electronic voting schemes should, ideally, comply with some essential requirements. Below we list those requirements, as defined in Chang & Lee (2006). We discuss, in a theoretical manner, the compliance of our proposed protocol with each of the requirements. (Such a discussion of compliance with essential requirements is common in papers dealing with secure electronic voting schemes, see e.g. Chung & Wu (2009); Li et al. (2008); Liaw (2004); Wu et al. (2014).)

Anonymity/Privacy. *The votes should remain anonymous throughout the entire process.* Our protocol achieves anonymity, as explained in Section 4.2. In particular, our protocol outputs only the desired election results, without revealing any information on the ballots.

Fairness. *A fair mechanism does not reveal intermediate results.* Our protocol is fair since all intermediate results (such as aggregated scores of candidates) are kept secret-shared; the protocol is designed to output just the identities of the winner(s) and nothing beyond that.

Convenience. *No special equipment is required and the voters do not need to learn any specialized technique.* Our protocol relies on basic and general-purpose cryptographic functions that can be found in many free libraries. In fact, as our protocol includes a mechanism for validating the cast ballots (Section 4.3), voters may vote from anywhere (and not necessarily from voting centers). They only need to download a simple software package that implements our protocol vis-a-vis the talliers.

Robustness. *No malicious intruder should be able to interrupt the procedure.* This requirement is addressed by the standard security mechanisms that should be implemented on top of our protocol, as discussed in Section 4.2: signing ballots before sending, verifying received ballots, and confirming the receipt and verification of each ballot.

Mobility. *The mechanism can be implemented to run on the World Wide Web.* Our protocol allows mobility since it requires no special equipment and it relies on common cryptographic toolkit.

Uniqueness. *Each voter is allowed to vote only once.* If a voter attempts to vote twice (Step 3 in Protocol 1), only his first vote will be processed, while the second one will be ignored. Also, one voter cannot shoot down a message from another voter (in order to prevent the latter voter's ballot to reach the talliers) since the voter expects to receive from each tallier a confirmation of receiving his signed and authenticated message. In case such a confirmation is not received, the voter can resend his message.

Completeness. *Only eligible voters are allowed to vote.* This requirement is achieved by the usage of certificated public keys and signatures.

Uncoercibility. *A voter must not be able to prove to a third party how he had voted, in order to prevent bribery.* The only way to prove to a third party W how a voter V_n had voted is if all D talliers send to W the shares of V_n's ballot $\mathbf{w}_n$. Hence, under our assumption that the tallier has an honest majority, our protocol offers uncoercibility.

Correctness. *Each ballot must be counted correctly.* Correctness follows directly from the talliers' semi-honesty (namely, that they follow the prescribed computations correctly). To ensure the talliers' semi-honesty in practical deployments, the talliers' software must be verified and authenticated, and it would be best to run it on a dedicated and tamper-proof machine (namely, a computer that is physically protected from any hacking attempts).

Efficiency. *The computational load of the whole process is required to be such that the result is obtained within a reasonable amount of time.* Our protocol complies with this requirement, as is evident from our cost analysis that shows that it is extremely lightweight (see Section 5).

The right to abstain. *A voter that wishes to abstain should be able to do so without revealing that fact to any other party (Gritzalis, 2002).* In all considered rules, each ballot is a vector and the final result is determined by the sum of the private vectors. Hence, a voter V_n who wishes to abstain can use $\mathbf{w}_n = \mathbf{0}$. Such an action is equivalent to not participating. Even the talliers cannot tell that a message received from some voter contains an all-zero vote, because of the secret sharing. Hence, our protocol allows confidential abstinence.

In view of the above, our proposed scheme is suitable for electronic elections of any kind and scale, from small-scale elections similar to the examples given in Section 1, to national elections in populations of any size, provided that the underlying voting rule is score-based. In national elections, it is possible to implement our electronic voting system in a manner that resembles existing voting systems. Citizens will arrive at the voting centers and identify using their stan-

dard identification card. They will then obtain access to a computer into which they will privately enter their selection. The software running the protocol will translate their selection into a ballot vector, as described in Section 3.1; subsequently, shares of that ballot will be distributed to the D talliers' servers, who will then proceed to process them as described in Section 4. As opposed to existing tallying systems, in our system, the *final and true* results will be determined with utmost certainty and security within seconds after the voting period ends.

7. Conclusion

We begin this section with a birds-eye summary of our study. Then, we identify the limitations of our method and provide an outlook on future research directions.

7.1. Summary

We considered a setting in which a group of voters wishes to elect K candidates out of a given list of candidates. We considered score-based rules and showed how to securely compute the winners using a secure multiparty protocol. Our protocol offers perfect ballot secrecy: apart from the desired output (the identity of the elected K candidates), all other information, such as the actual ballots or the aggregated scores that each of the candidates received, is kept secret from all parties – voters and talliers alike. (As indicated in Section 4.2, the protocol herein allows the talliers to deduce the ranking among the candidates, but a small modification of the underlying circuit may hide even that information.)

Such level of privacy may be essential in some scenarios. For example, when a prize committee needs to select K recipients out of M candidates, it is desirable to determine only the identity of the K prize recipients without revealing their internal ranking or their aggregated scores, as such pieces of information might expose undesired information about the ballots.

Such perfect secrecy may increase the confidence of the voters in the voting system, so that they would be encouraged to participate in the elections and vote truthfully without fearing that their private vote would be disclosed to anyone else. Furthermore, our technique can be used during iterative voting (Airiau & Endriss, 2009; Lev & Rosenschein, 2016; Meir et al., 2010; Dery et al., 2019). If, during iterative voting, the candidates' scores are kept secret from all parties, then voters would not obtain from the voting process information that could have been used for strategic voting. (Of course, the perfect secrecy of our protocol

21

cannot stop strategic voting altogether, since voters may still base strategic voting on information from polls, rumors and other communication channels.)

The protocol complies with conventional security desiderata, as discussed in Section 6. An analysis of its computational and commutation costs shows that it is practical as it is extremely lightweight. The protocol is based solely on existing cryptographic arsenal. This is a prominent advantage of our protocol; indeed, protocols that can be implemented on top of existing libraries are advantageous over protocols that require the development, scrutiny and assimilation of new cryptographic components and, therefore, might be unattractive to practitioners.

7.2. Limitations and future work

Our method has several limitations, which suggest corresponding future research endeavours to resolve them.

(a) **Convincing the public.** As with each new technology, our model faces the difficulty of "selling" it to the public and the legislators, and convincing them of its safety and other desired advantages. As a first step in this direction, a wide-range user study should be carried out.

(b) **Extension to order-based rules.** Herein we focused on score-based rules. Another important family of rules consists of rules where the voter submits an ordered list of preferences. Such rules are called *order-based* or *pairwise-comparison* voting rules (see Brandt & Sandholm (2005)). Two prominent rules in this family are – COPELAND (Copeland, 1951) and MAX-IMIN (a.k.a KRAMER-SIMPSON) (Kramer, 1977; Simpson, 1969). We intend to devise a protocol for securely computing the winners in voting systems that are based on such rules.

(c) **Extension to multi-winner elections.** Our current protocol can select the top K candidates, for any $K < M$; namely, it can output the K candidates that got the highest aggregated scores. However, there are multi-winner election protocols that are designed specifically for selecting the K candidates that would satisfy the voters the most (Faliszewski et al., 2017; Elkind et al., 2017), 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 protocol. Examples for voting rules that are designed for such a purpose are CHAMBERLIN-COURANT (Chamberlin & Courant, 1983) and MONROE (Monroe, 1995).

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

(e) **Robustness.** In our protocol, all ballots are shared between all D talliers using a simple D-out-of-D secret sharing scheme (see Section 3.2). As a result, our protocol is very secure, since the ballots are perfectly secure, unless all D talliers collude and recover the ballots by combining all of their shares. However, another implication of using such a secret sharing scheme is that even if a single tallier, say T_D, becomes dysfunctional (e.g., due to a physical damage to its servers, or a cyberattack) then all ballot information is lost and the voting procedure should be performed again with $T_1, \ldots, T_{D-1}, T_D'$, where T_D' is another new tallier, or T_D' is T_D after it had recovered. In order to add robustness to our system, a Shamir D'-out-of-D secret sharing scheme (Section 3.2) should be used instead. In that case, even if any subset of $D - D'$ talliers become dysfunctional, then the remaining D' talliers can still complete the computation of the final voting results. In order to do that, our MPC protocol should be modified in order to be compliant with such secret sharing.

(f) **Restraining malicious talliers.** Our protocol assumes that the talliers are semi-honest, i.e., they follow the prescribed computations correctly. As discussed in Section 6, the semi-honesty of the talliers can be ensured in practice by securing the software and hardware of the talliers. However, one can provide even a stronger protection shield by cryptographic means. Namely, it is possible to design an MPC protocol that would be immune even to malicious talliers that may attempt to infer secret ballot information or intermediate voting results, or may even try to affect the final voting results. MPC protocols that are designed to be immune to malicious parties are usually significantly costlier and more complex than the corresponding MPC protocols for semi-honest parties. However, since in common voting systems, the final results are published usually hours and even days after voting had ended, the implementation of costlier MPC protocols that are designed to resist malicious talliers is not expected to be problematic, and it will enhance even further the security of the system and the trust of voters

in the preservation of their privacy.

To conclude, the secure voting protocol that we presented here is of relevance to any setting in which voting should be implemented in a manner that ensures perfect privacy. The protocol promotes truthful voting and the final results are obtained within seconds. Even though the presented protocol is relevant and adequate as is to a wide range of practical voting scenarios, it opens up several interesting research directions that could widen its application scope and further strengthen its security.

Funding

References

Airiau, S., & Endriss, U. (2009). Iterated majority voting. In *International Conference on Algorithmic DecisionTheory* (pp. 38–49).

Alwen, J., Shelat, A., & Visconti, I. (2008). Collusion-free protocols in the mediated model. In *CRYPTO* (pp. 497–514).

Balinski, M., & Laraki, R. (2007). A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences*, *104*, 8720–8725.

Benaloh, J. (1986). Secret sharing homomorphisms: Keeping shares of a secret secret. In *CRYPTO* (pp. 251–260).

Benaloh, J. (1987). *Verifiable Secret-Ballot Elections*. Technical Report MSR-TR-1987-1.

Benkaouz, Y., & Erradi, M. (2015). A distributed protocol for privacy preserving aggregation with non-permanent participants. *Computing*, *97*, 893–912.

Blakley, G. (1979). Safeguarding cryptographic keys. In *International Workshop on Managing Requirements Knowledge* (pp. 48:313–317).

Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (2016). *Handbook of computational social choice*. Cambridge University Press.

Brandt, F., & Sandholm, T. (2005). Decentralized voting with unconditional privacy. In *AAMAS* (pp. 357–364).

Canard, S., Pointcheval, D., Santos, Q., & Traoré, J. (2018). Practical strategy-resistant privacy-preserving elections. In *European Symposium on Research in Computer Security* (pp. 331–349).

Chamberlin, J. R., & Courant, P. N. (1983). Representative deliberations and representative decisions: Proportional representation and the borda rule. *The American Political Science Review*, (pp. 718–733).

Chang, C.-C., & Lee, J.-S. (2006). An anonymous voting mechanism based on the key exchange protocol. *Computers & Security*, *25*, 307–314.

Chaum, D. (1982). Blind signatures for untraceable payments. In *CRYPTO* (pp. 199–203).

Chaum, D. (1988). Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *EUROCRYPT* (pp. 177–182).

Chen, C.-L., Chen, Y.-Y., Jan, J.-K., Chen, C.-C. et al. (2014). A secure anonymous e-voting system based on discrete logarithm problem. *Applied Mathematics & Information Sciences*, *8*, 2571–2578.

Chen, L., Xu, L., Xu, S., Gao, Z., & Shi, W. (2019). Election with bribe-effect uncertainty: A dichotomy result. In *IJCAI* (pp. 158–164).

Chida, K., Genkin, D., Hamada, K., Ikarashi, D., Kikuchi, R., Lindell, Y., & Nof, A. (2018). Fast large-scale honest-majority MPC for malicious adversaries. In *CRYPTO* (pp. 34–64).

Chung, Y.-F., & Wu, Z.-Y. (2009). Approach to designing bribery-free and coercion-free electronic voting scheme. *Journal of Systems and Software*, *82*, 2081–2090.

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

Corò, F., Cruciani, E., D'Angelo, G., & Ponziani, S. (2019). Exploiting social influence to control elections based on scoring rules. In *IJCAI* (pp. 201–207).

Cramer, R., Gennaro, R., & Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT* (pp. 103–118).

Damgård, I., Jurik, M., & Nielsen, J. B. (2010). A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, *9*, 371–385.

Dery, L., Obraztsova, S., Rabinovich, Z., & Kalech, M. (2019). Lie on the fly: Strategic voting in an iterative preference elicitation process. *Group Decision and Negotiation*, *28*, 1077–1107.

Dey, P., Misra, N., Nath, S., & Shakya, G. (2019). A parameterized perspective on protecting elections. In *IJCAI* (pp. 238–244).

Elkind, E., Faliszewski, P., Laslier, J.-F., Skowron, P., Slinko, A., & Talmon, N. (2017). What do multiwinner voting rules do? an experiment over the two-dimensional euclidean domain. In *AAAI*.

Elkind, E., Gan, J., Obraztsova, S., Rabinovich, Z., & Voudouris, A. A. (2019). Protecting elections by recounting ballots. In *IJCAI* (pp. 259–265).

Faliszewski, P., Skowron, P., Slinko, A., & Talmon, N. (2017). Multiwinner voting: A new challenge for social choice theory. *Trends in computational social choice*, *74*, 27–47.

Fujioka, A., Okamoto, T., & Ohta, K. (1992). A practical secret voting scheme for large scale elections. In *AUSCRYPT* (pp. 244–251).

Gritzalis, D. A. (2002). Principles and requirements for a secure e-voting system. *Computers & Security*, *21*, 539–556.

Ibrahim, S., Kamat, M., Salleh, M., & Aziz, S. R. A. (2003). Secure e-voting with blind signature. In *NCTT* (pp. 193–197).

Kramer, G. H. (1977). A dynamical model of political equilibrium. *Journal of Economic Theory*, *16*, 310–334.

Lev, O., & Rosenschein, J. S. (2016). Convergence of iterative scoring rules. *Journal of Artificial Intelligence Research*, *57*, 573–591.

Li, C.-T., Hwang, M.-S., & Liu, C.-Y. (2008). An electronic voting protocol with deniable authentication for mobile ad hoc networks. *Computer Communications*, *31*, 2534–2540.

Liaw, H.-T. (2004). A secure electronic voting protocol for general elections. *Computers & Security*, *23*, 107–119.

Masthoff, J. (2011). Group recommender systems: Combining individual models. In *Recommender systems handbook* (pp. 677–702). Springer.

Meir, R., Polukarov, M., Rosenschein, J. S., & Jennings, N. R. (2010). Convergence to equilibria in plurality voting. In *AAAI*.

Monroe, B. L. (1995). Fully proportional representation. *American Political Science Review*, (pp. 925–940).

Nair, D. G., Binu, V. P., & Kumar, G. S. (2015). An improved e-voting scheme using secret sharing based secure multi-party computation. *CoRR*, *abs/1502.07469*.

Nishide, T., & Ohta, K. (2007). Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *PKC* (pp. 343–360).

Nurmi, H. (2012). *Comparing voting systems* volume 3. Springer Science & Business Media.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT* (pp. 223–238).

Park, C., Itoh, K., & Kurosawa, K. (1993). Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT* (pp. 248–259).

Peng, K., Aditya, R., Boyd, C., Dawson, E., & Lee, B. (2004). Multiplicative homomorphic e-voting. In *INDOCRYPT* (pp. 61–72).

Sako, K., & Kilian, J. (1995). Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In *EUROCRYPT* (pp. 393–403).

Shamir, A. (1979). How to share a secret. *Commun. ACM*, *22*, 612–613.

Simpson, P. B. (1969). On defining areas of voter choice: Professor tullock on stable voting. *The Quarterly Journal of Economics*, (pp. 478–490).

Wu, Z.-Y., Wu, J.-C., Lin, S.-C., & Wang, C. (2014). An electronic voting mechanism for fighting bribery and coercion. *Journal of Network and Computer Applications*, *40*, 139–150.

Yang, Y. (2019). Complexity of manipulating and controlling approval-based multiwinner voting. In *IJCAI* (pp. 637–643).

Yao, A. (1982). Protocols for secure computation. In *FOCS* (pp. 160–164).

Zagórski, F., Carback, R. T., Chaum, D., Clark, J., Essex, A., & Vora, P. L. (2013). Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *ACNS* (pp. 441–457).