A Smart Contract System for Decentralized Borda Count Voting

Somnath Panja, Samiran Bag, Feng Hao and Bimal Roy

***Abstract*—In this paper, we propose the first self-tallying decentralized e-voting protocol for a ranked-choice voting system based on Borda count. Our protocol does not need any trusted setup or tallying authority to compute the tally. The voters interact through a publicly accessible bulletin board for executing the protocol in a way that is publicly verifiable. Our main protocol consists of two rounds. In the first round, the voters publish their public keys, and in the second round they publish their randomized ballots. All voters provide Non-interactive Zero-Knowledge (NIZK) proofs to show that they have been following the protocol specification honestly without revealing their secret votes. At the end of the election, anyone including a third-party observer will be able to compute the tally without needing any tallying authority. We provide security proofs to show that our protocol guarantees the maximum privacy for each voter. We have implemented our protocol using Ethereum's blockchain as a public bulletin board to record voting operations as publicly verifiable transactions. The experimental data obtained from our tests show the protocol's potential for the real-world deployment.**

***Index Terms*—E-voting, Self-enforcing voting, NIZK proofs, Borda count voting, Ethereum smart contract, Blockchain technology.**

I. Introduction

In a Borda count voting system, the voters cast their vote by ranking the candidates according to the order of preference. Each candidate obtains some points according to her position in the ranking done by a particular voter. In the end all the points obtained by her from all the voters are summed up and on the basis of this sum the winner is selected. For example, the least preferred candidate may get 0 point, the next one may get 1 point and so on. The Borda count voting system has been employed in the elections in Nauru [51], Slovenia [26] and in Kiribati [51]. In Ireland, a modified version of the Borda count system has been used by the Green party to elect its president [22]. Unlike the plurality voting system, the Borda count systems are designed to gather more information from the voter regarding her predilection toward more than one candidate.

Borda count voting using traditional paper ballot is not only time consuming, but also prone to human errors. Hence, there is a desirable need to advance toward using an electronic voting system. However, with the advent of e-voting systems comes the need to ensure privacy and integrity of the voting system. An electronic system can be vulnerable to several attacks like intrusion, software alteration/modification, eavesdropping etc.

The lack of assurance on the integrity of e-voting systems has encouraged researchers to devise e-voting systems that provide end-to-end verifiability and that are proven to be secure. The research on end-to-end verifiable e-voting systems started with the pioneering work by Chaum [15]. Chaum's scheme uses visual cryptography to protect the privacy of voters. Every voter is issued with two strips of paper corresponding to a vote. Each one of the two strips does not divulge any secret on their own. When the two strips are superposed on one another under a custom viewfinder, the vote is revealed. The voter retains one strip and the other one is digitized before getting destroyed. Once the polling station voting has finished, all the saved voter receipts are published on the bulletin board, so that the voters can verify that their ballots are not discarded. This work first highlighted the notion of end-to-end verifiability in voting. A voting system is called end-to-end verifiable if it ensures that 1) every vote is cast as intended, 2) every vote is recorded as cast, and 3) every vote is tallied as recorded. Some other notable research works in this area are MarkPledge [48], Prêt à Voter [53], Punchscan [24], Scantegrity [14], Scantegrity II [16], scratch & vote [3], STAR-Vote [6], Adder [37], and Helios [4]. These systems use either mix-net [15] or homomorphic encryption [2], but they all involve a set of trustworthy tallying authorities (TAs) to perform the decryption and tallying process in a publicly verifiable way.

A major difficulty of implementing the above schemes is to find and manage a set of trustees who perform complex cryptographic operations as tallying authorities. Threshold control schemes can be applied to distribute the trust among TAs. Nonetheless, if a sufficient number of trustees collude, they can trivially breach the privacy of the e-voting system.

Hao, Ryan and Zieliński proposed a decentralized online voting scheme in [30]. This scheme allows a finite number of voters to conduct voting without requiring the help of any tallying authorities. This scheme is called Open-Vote network (OV-net). OV-Net consists of two rounds. In the first round each voter publishes her public key on the public bulletin board. In the second round each voter publishes her randomized ballot that is generated using the public keys of other voters, the secret key of that voter and her secret vote. Once all the encrypted ballots are available on the bulletin board, anyone can easily calculate the tally from them. This scheme relies on non-interactive zero-knowledge proofs for proving the well-formedness of each ciphertext. The scheme offers the maximum possible privacy guarantee as each voter

S. Panja and B. Roy are with the Applied Statistics Unit, Indian Statistical Institute, Kolkata, India. E-mails: {somn.math2007@gmail.com, bimal@isical.ac.in}. S. Bag and F. Hao are with the Department of Computer Science, University of Warwick, United Kingdom. E-mails: {samiran.bag, feng.hao}@warwick.ac.uk. This work is supported by the Royal Society grant, ICA/R1/180226.

learns nothing more than the tally and their own vote. A public observer learns nothing more than the tally from the bulletin board.

The OV-Net scheme is designed to support plurality voting where every voter gets to vote for a single candidate. In this paper, we propose a decentralized Borda count e-voting scheme by extending OV-Net to support ranking-based voting. Similar to the OV-Net scheme, our Borda count scheme also has two rounds. In the first round, the voters publish their public keys, and in second round they publish their encrypted ballots under a public key re-constructed by combining every other voter's public keys. Each encrypted ballot comes with a Non-interactive Zero-Knowledge (NIZK) proof to prove the well-formedness of the ballot. Once all the voters submit their ballots, the tally can be computed from the published information available on the bulletin board. Anyone can compute the tally and verify the correctness of all operations with the help of the NIZK proofs. Our scheme offers strong privacy guarantee. A probabilistic polynomial time adversary learns nothing other than the tally and whatever she can interpret from the tally. We have implemented the Borda count e-voting scheme on Ethereum's [64] platform and have evaluated the efficiency of our scheme.

Contributions. Our contributions in this paper include the following. (1) We propose the first self-tallying decentralized Borda count voting protocol. The proposed protocol provides maximum voter privacy: an individual vote can only be revealed by a full-collusion attack that involves all other voters. All voting data is publicly available, and the correct execution of the protocol can be verified by any public observer. The proposed protocol does not require any trusted authority to compute the tally; the tally can be computed by each voter, as well as by any observer of the election. (2) We provide security proofs to prove the security of the proposed scheme. In particular, we show that the proposed scheme guarantees the maximum voter privacy against colluding voters. (3) We provide an implementation of the proposed protocol over Ethereum Blockchain. It is a boardroom-scale voting system implemented as a smart contract in Ethereum. Our implementation demonstrates the feasibility of using Ethereum for secure Borda count voting with public verifiability.

II. Related work

In most of the e-voting systems, trustworthy election authorities are required to preserve voter's privacy, to decrypt the vote and to compute the tally in a verifiable manner. Generally, threshold cryptography is used to distribute this trust among multiple tallying authorities; see, for example, Helios [1]. However, while using threshold cryptography, if the tallying authorities collude among themselves altogether, voter's privacy will be lost.

Several researchers have proposed e-voting systems based on blockchain. In [66], Zhao and Chan propose a voting system using Bitcoin. In their voting system, random numbers and zero-knowledge proofs are used to hide the vote. In [60], Tarasov and Tewari propose an e-voting system based on cryptocurrency. In this system, a centralised trusted authority exists to coordinate the election. Tivi [63], Followmyvote [25] and The Blockchain Voting Machine [32] are Internet voting systems that use blockchain as a ballot box. These systems depend on trusted authorities to achieve voter's privacy. In Tivi, the trusted authority shuffles the encrypted votes before decrypting and computing the tally. In Followmyvote, the trusted authority obfuscates the link between a voter's identity and her voting key before the voter casts her vote. In our proposed protocol, the voter's privacy and the tally procedure do not depend on trusted election authorities. We implement the proposed protocol using smart contract in such a way that the Ethereum bockchain's consensus mechanism enforces the execution of the voting protocol. Recently, the Abu Dhabi Securities Exchange [33] has launched a blockchain-based voting service. In Estonia, blockchain-based voting systems [9] have been proposed for the internal elections of political parties and shareholder voting. The possibility of using blockchain in e-voting is also discussed in a report [9] by the Scientific Foresight Unit of the European Parliamentary Research Service. Recently, in [5], Bag et al. propose an end-to-end verifiable Borda count voting system. However, their scheme is on a centralised setting where a central facility (i.e., a touch-screen voting machine) is used to directly record votes from voters. In such a setting, it is inevitable that the touch-screen machine learns the voter's choice. In this paper, we propose the first self-tallying decentralized Borda count voting protocol, in which voters cast votes using their own devices in a distributed manner. No third-party entity can learn the voter's input unless all other voters are compromised (i.e., in a full-collusion attack).

The first self-tallying voting protocol was proposed by Kiayias and Yung [38] for boardroom voting. Their protocol has the following three attractive features: it is self-tallying; it provides the maximum voter privacy; and it is dispute-free. We discuss these properties in detail in a later section. Their protocol executes in three rounds. However, the computational load for each voter is heavy and it increases linearly with the number of voters. A subsequent protocol by Groth [29] improves the computational complexity. The computational load for each voter is less than the Kiayias and Yung's protocol [38] and remains constant with the number of voters; however, the protocol trades off round efficiency for less computation. It requires $(n+1)$ rounds, where n is the number of voters. The round efficiency is worse than the Kiayias and Yung's protocol [38]. Hao, Ryan and Zieliński investigated the computation complexity and proposed the Open Vote Network (OV-Net) protocol [30]. Their protocol significantly improves computational complexity. Their protocol executes in only two rounds. In fact, their protocol is a generalization of the anonymous veto network (AV-net) protocol [31] with the added self-tallying function. McCorry et al. [46] provided the first implementation of the OV-Net protocol using the Ethereum blockchain.

III. Preliminaries

In this section, we describe the security definitions that our proposed protocol is expected to satisfy. We also state the assumptions based on which we prove these security properties.

A. Desirable properties

In a decentralised voting system, some voters may collude with each other to breach other voters' privacy or manipulate the voting outcome. A full collusion against a particular voter occurs when all other voters involve in the collusion. No decentralized system can preserve an honest voter's privacy in case of full collusion since the honest voter's vote can be obtained by subtracting the colluding voters' votes from the final tally. Therefore, we only consider partial collusion which involves some voters, but not all.

Under the threat model of partial collusion, the following three properties (also see [38], [29]) should be fulfilled by a decentralized voting protocol.

1) *Maximum ballot secrecy:* This is an extension of the usual ballot secrecy requirement. In a voting system with maximum ballot secrecy, an attacker who colludes with a group of voters will only learn the partial tally of the remaining voters, but nothing beyond that.
2) *Self-tallying:* During the tallying phase, the final tally can be computed by anyone including voters and third-party observers without external help. This is naturally expected in a decentralized voting protocol.
3) *Dispute-freeness:* A voting scheme is dispute-free if every observer of the election can verify the fact that every voter follows the protocol honestly. This requirement means that the result should be publicly verifiable.

In a self-tallying voting protocol, the last voter can compute the tally before casting her vote. This leads to two issues. First, the last voter can use the knowledge of the tally to decide how she will cast her vote. This issue can potentially influence the result of the election. To prevent this issue, Kiayias and Yung [38] and Groth [29] suggest that an election authority can cast the last vote. During the tallying phase, this last vote is excluded from the final tally. We can apply this method in our implementation, however, in this case, the election authority needs to be trusted not to collude with the last voter. Therefore, this method relies on the trusted election authority. Instead, McCorry et al. [46] propose an extra commitment round to address this issue. In this round, every voter stores the hash of their encrypted vote in the blockchain as a commitment. The last voter will still be able to compute the tally before casting her vote, however, she will not be able to change her vote. We follow the same approach.

The second issue that since the last voter knows the tally before casting her vote, she may refrain from casting her vote if she is dissatisfied with the result of the election. In that case, no one will be able to compute the final tally, and the election will need to be restarted. To circumvent this issue, Kiayias and Yung [38] and Khader et al. [36] propose an additional round engaging the rest of the voters. However, in this case, we must assume the remaining voters do not drop out of the election half-way; otherwise, the election will be aborted again. To address this issue, we use Ethereum's blockchain and smart contract to enforce a financial incentive for all voters using a deposit and refund paradigm as done in [46]. In our implementation, it is mandatory for all voters to deposit some money into the smart contract to register for an election. This deposit is refunded automatically to the voter once her vote is successfully accepted by the blockchain. A voter who registers for an election but withholds her vote simply loses the deposit. This provides a countermeasure to this abortive issue. The confiscated deposit could be used as a compensation for all compliant voters or be donated to a charity.

B. Cryptographic assumption

We state the cryptographic assumption that we use to prove the security properties of our proposed protocol. If this assumption holds, the proposed protocol satisfies the above security properties. We first describe some notations that we use throughout our paper.

Notation: We follow the notation introduced by Camenisch and Stadler [12]. We use $P_K\{\lambda : \Gamma = \gamma^\lambda\}$ to denote a non-interactive proof of knowledge of a secret λ such that $\Gamma = \gamma^\lambda$ for publicly known Γ and γ. We shorten the notation to $P_K\{\lambda\}$ if the context is clear.

Cryptographic setup: Our system works on a DSA like multiplicative cyclic group setting or an ECDSA-like group setting over an elliptic curve, where the decision Diffie-Hellman problem is assumed to be intractable [19]. Let $\mathbb{G}_q$ be a subgroup of Z_p^* of prime order q, where $q \,|\, p-1$. Let g be a generator of that group.

The decision Diffie-Hellman assumption [19] is defined as follows:

Assumption 1: (DDH) If α, β are randomly and uniformly chosen from $\mathbb{Z}_q^*$, given $(g, g^\alpha, g^\beta, \rho)$ where $\rho \in \{g^{\alpha\beta}, R\}$ and R is randomly and independently chosen from $\mathbb{G}_q$, it is hard to decide whether $\rho = g^{\alpha\beta}$ or $\rho = R$.

IV. Our scheme

The Borda count scheme is a ranked choice voting method in which voters rank candidates in order of preference. A score is associated with each rank. Let there are k candidates competing in an election. Assume a score a_j is associated with the j-th rank and $a_{j-1} > a_j, \forall j \in \{2, ..., k\}$, i.e., a higher score implies a higher rank of the candidate. At the end of the election, the scores obtained by a candidate are added together. Finally, the candidates are ranked according to their scores obtained. The candidate with the highest score wins the election. In this scheme, a participant's vote will be of the form $(v_1, v_2, ..., v_k)$, where $(v_1, v_2, ..., v_k)$ is a permutation of $(a_1, a_2, ..., a_k)$.

In our protocol, we assume that there is an authenticated public channel available for each participant. This assumption is common in previous e-voting protocols; see, for example, Kiayias and Yung's protocol [38], Groth's protocols [29], the open vote network [30], and general multi-party secure computation protocols [27], [28]. This authenticated public channel can be realized by using physical means or a public bulletin board where recorded ballots are stored securely in an append only manner [46].

Our protocol is inspired by Open Vote Network [30] but we have adapted the protocol to support Borda count. This results

in a new e-voting protocol, which is also the first ranked-choice voting system based on Borda count in a decentralized setting.

Our protocol works in two rounds. Assume that there are n participants in an election. We denote the i-th participant as V_i. They all agree on public group parameters $(\mathbb{G}_q, g)$. Let there are k candidates competing in the election. Let us assume that the score corresponding to the j-th rank is a_j, where $a_{j-1} > a_j; \forall j \in \{1, 2, ..., k\}$.

A. Voting Phase

Each participant V_i generates k random values $(x_{i1}, x_{i2}, ..., x_{ik})$ as their secrets, where $x_{ij} \in_R \mathbb{Z}_q; \forall j \in \{1, 2, ..., k\}$. Each participant executes the following two-round protocol. We denote the vote cast by the i-th participant V_i as $v_i = (v_{i1}, v_{i2}, ..., v_{ik})$, where $(v_{i1}, v_{i2}, ..., v_{ik})$ is a permutation of $(a_1, a_2, ..., a_k)$. Here, v_{ij} is the score given by the participant V_i to the j-th candidate.

First round. Every participant V_i calculates $X_{i1} = g^{x_{i1}}, X_{i2} = g^{x_{i2}}, ..., X_{ik} = g^{x_{ik}}$ and publishes $(X_{i1}, P_K\{x_{i1}\}, X_{i2}, P_K\{x_{i2}\}, \ldots, X_{ik}, P_K\{x_{ik}\})$, where $P_K\{x_{ij}\}$ is the non-interactive zero-knowledge (NIZK) proof for $x_{ij}; \forall j \in \{1, 2, ..., k\}$. The NIZK proofs are generated by using Schnorr's signature [54] (see Appendix A for more details).

At the end of this round, every participant verifies the validity of all zero-knowledge proofs. Each participant V_i then computes $g^{y_{ij}} = \prod_{l=1}^{i-1} g^{x_{lj}} / \prod_{l=i+1}^{n} g^{x_{lj}}; \forall j \in \{1, 2, \ldots, k\}$.

Second round. Each participant V_i calculates $Z_{ij} = \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}}; \forall j \in \{1, 2, ..., k\}$ and generates a non-interactive zero-knowledge (NIZK) proof to prove the well-formedness of the ballot. The NIZK associated with each participant's ballot proves that $(v_{i1}, v_{i2}, ..., v_{ik})$ is a permutation of $(a_1, a_2, .., a_k)$. In order to prove the statement, it is sufficient to prove the following k relations.

- $a_1 \in \{v_{i1}, v_{i2}, , ..., v_{ik}\}$ for the score corresponding to the 1-st rank.
- $a_2 \in \{v_{i1}, v_{i2}, , ..., v_{ik}\}$ for the score corresponding to the 2-nd rank.
- ...
- $a_k \in \{v_{i1}, v_{i2}, , ..., v_{ik}\}$ for the score corresponding to the k-th rank.

The above k relations hold true if and only if the following relations are true:

$\vee_{j=1}^{k}(X_{ij} = g^{x_{ij}} \wedge Z_{ij} = \{g^{y_{ij}}\}^{x_{ij}} g^{a}); \forall a \in \{a_1, a_2, ..., a_k\}$. We denote this NIZK proof as $\Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, .., k\}$, where $X_i = (X_{i1}, X_{i2}, ..., X_{ik}), Z_i = (Z_{i1}, Z_{i2}, ..., Z_{ik})$. More details about the NIZK proof can be found in Appendix A.

Each participant V_i publishes $(Z_{i1}, Z_{i2}, ..., Z_{ik})$ and k NIZK proofs $\Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\}$.

B. Tallying phase

Anyone can compute $\prod_{i=1}^{n} \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}} = g^{\Sigma_{i=1}^{n} v_{ij}}$, where $j \in \{1, 2, ..., k\}$. This equality follows from the fact that $\Sigma_{i=1}^{n} x_{ij} y_{ij} = 0; \forall j \in \{1, 2, ..., k\}$ (proposition 1, see also [31]).

Each participant, as well as observers of the election, can check the validity of all the NIZK proofs to ensure that no badly formed vote has been cast to distort the tally.

The total score obtained by the j-th candidate is $\Sigma_{i=1}^{n} v_{ij}$, which is normally a small number for all $j \in \{1, 2, ..., k\}$. Since the quantity $\Sigma_{i=1}^{n} v_{ij}$ is a small number, the discrete logarithm of $g^{\Sigma_{i=1}^{n} v_{ij}}$ can be computed by exhaustive search or Shanks' baby-step giant-step algorithm [43]. Finally, each participant and all observers of the protocol can rank candidates in order of their total scores.

Proposition 1: For x_{ij} and y_{ij} as defined above $\Sigma_{i=1}^{n} x_{ij} y_{ij} = 0; \forall j \in \{1, 2, ..., k\}$.

Proof: According to the protocol, $y_{ij} = \Sigma_{l<i} x_{lj} - \Sigma_{l>i} x_{lj}; \forall j \in \{1, 2, ..., k\}$.

Now for any fix $j \in \{1, 2, ..., k\}$, $\Sigma_{i=1}^{n} x_{ij} y_{ij} = \Sigma_{i=1}^{n} x_{ij} (\Sigma_{l<i} x_{lj} - \Sigma_{l>i} x_{lj}) = 0$. □

The use of NIZK proofs in the protocol is to ensure that all participants follow the protocol faithfully. In the first round, each participant posts k NIZK proofs to prove her knowledge of the exponents $(x_{i1}, x_{i2}, ..., x_{ik})$. The Fiat-Shamir heuristic is employed in our protocol to construct NIZK proofs [23]. Consequently, our NIZK proofs are in the Random Oracle Model [7]. The algorithm 1 and algorithm 2 provided in Appendix A describe the procedure to create and verify these NIZK proofs respectively.

In the second round, each participant posts k NIZK proofs to prove that her encrypted vote is a permutation of $(a_1, a_2, ..., a_k)$ without reveling which permutation. In order to prove this, first note that the terms of our protocol form exponential ElGamal encryptions of v_{ij}, where $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., k\}$. This can be realized by treating $g^{y_{ij}}$ as the public key and using the published terms of 1-st round. Thus, we form

$(g^{x_{ij}}, \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}})$, where $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., k\}$.

This will be an ElGamal encryption of $g^{v_{ij}}; \forall j \in \{1, 2, ..., k\}$ with public key $g^{y_{ij}}$ and randomization $x_{ij}; \forall j \in \{1, 2, ..., k\}$. Thus, the published terms for the first and second rounds form an exponential ElGamal encryption of v_{ij}, $\forall i \in \{1, 2, ..., n\}$ and $\forall j \in \{1, 2, ..., k\}$. We use an efficient NIZK proof technique proposed by Cramer, Damgård, and Schoenmakers [17] to construct proofs of conjunctive knowledge, disjunctive knowledge and combination of both. This essentially proves that $(v_{i1}, v_{i2}, ..., v_{ik})$ is a permutation of $(a_1, a_2, ..., a_k)$ without revealing which permutation (i.e. which message corresponds to which ciphertext). Note that we do not need to decrypt these ciphertexts in order to obtain the tally. The algorithm 3 and algorithm 4 provided in Appendix A describe the procedure to create and verify these 1-out-of-k NIZK proofs respectively.

V. Security analysis

In this section, we prove that our scheme is secure against all probabilistic polynomial adversaries who try to deduce the vote given by a voter. We also show that the public bulletin board does not reveal any additional information regarding the voter's privacy other than the tally. In case of partial collusion,

we prove that if an attacker colludes with some m number of voters, then she will learn the partial tally of the $n-m$ voters, but nothing beyond that. The partial tally of the honest voter's vote can be obtained by subtracting the colluding voter's vote from the final tally. Therefore, colluding voters can always compute the partial tally of votes of the remaining voters. The security of our scheme relies on the intractability of the Decisional Diffie-Hellman (DDH) problem. We prove that if the DDH problem is intractable in the group $\mathbb{G}_q$, then this scheme is secure.

In the following sections, we show that our protocol satisfies the three security requirements mentioned in section III-A.

A. Maximum ballot secrecy

In this section, we show that our protocol is secure under the threat model of partial collusion. In the protocol, each participant V_i sends k ephemeral public keys $(g^{x_{i1}}, g^{x_{i2}}, ..., g^{x_{ik}})$ along with k NIZK proofs of exponents in the first round and sends an encrypted ballot $(g^{y_{i1}x_{i1}}g^{v_{i1}}, g^{y_{i1}x_{i2}}g^{v_{i2}}, ..., g^{y_{i1}x_{ik}}g^{v_{ik}})$ with k 1-out-of-k NIZK proofs in the second round. The k 1-out-of-k NIZK proofs in the second round ensure that $(v_{i1}, v_{i2}, ..., v_{ik})$ is a permutation of $(a_1, a_2, ..., a_k)$. In this protocol, the value of y_{ij} depends on the values of secret keys x_{ij} of all voters except V_i, where $j \in \{1, 2, ..., k\}$. We now discuss the security properties of y_{ij}. Let us consider any participant V_i, where $i \in \{1, 2, ..., n\}$. The following properties hold true for any participant V_i.

Lemma 1: Considering the threat model of partial collusion against a participant V_i, the y_{ij} is a secret random value in $\mathbb{Z}_q$ to attackers for all $j \in \{1, 2, ..., k\}$.

Proof: Let us first fix any $j \in \{1, 2, ..., k\}$. Since we are considering the partial collusion against V_i, there exists at least one other participant V_m $(m \neq i)$ who is not involved in the collusion. Hence, the secret random value x_{mj} generated by the participant V_m is unknown to the colluders. According to the protocol, x_{mj} is uniformly distributed over $\mathbb{Z}_q$ and known only to the participant V_m. Note that for a fixed j, y_{ij} is computed from all x_{ij} known to colluders (in the worst case) and a random number x_{mj} unknown to the colluders. Therefore, y_{ij} is uniformly distributed over $\mathbb{Z}_q$ and unknown to the colluders. Since the above discussion holds true for any $j \in \{1, 2, ..., k\}$, y_{ij} is a secret random value in $\mathbb{Z}_q$ to attackers for all $j \in \{1, 2, ..., k\}$ even in the worst case. □

Lemma 2: If the *assumption 1 (DDH)* holds in the group $\mathbb{G}_q$, then given $g, g^{\alpha_1}, g^{\alpha_2}, ..., g^{\alpha_m}$ and $g^{\beta_1}, g^{\beta_2}, ..., g^{\beta_m} \in \mathbb{G}_q$, and a challenge $\rho \in \{\rho_1, \rho_2\}$, where $\rho_1 = (g^{\alpha_1\beta_1}, g^{\alpha_2\beta_2}, ..., g^{\alpha_m\beta_m})$ and $\rho_2 = (R_1, R_2, ..., R_m), R_i \in \mathbb{G}_q, \forall i \in \{1, 2, ..., m\}$, it is hard to decide whether $\rho = \rho_1$ or $\rho = \rho_2$.

Proof: We prove the lemma by showing that if there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that can decide whether $\rho = \rho_1$ or $\rho = \rho_2$, then we can use the same to construct another adversary $\mathcal{B}$ against *assumption 1*. We construct the adversary $\mathcal{B}$ as follows. The inputs to the adversary $\mathcal{B}$ are g, g^{α}, g^{β}. The adversary $\mathcal{B}$ receives the challenge $\rho = \{g^{\alpha\beta}, R\}$, where $R \in_R \mathbb{G}_q$. It chooses $\gamma_1, \gamma_2, ..., \gamma_m$ and $\eta_1, \eta_2, ..., \eta_m$ uniformly at random from $\mathbb{Z}_q$. It computes $\delta_i = \rho^{\gamma_i\eta_i}, \forall i \in \{1, 2, ..., m\}$. $\mathcal{B}$ calculates $g^{\beta_i} = g^{\beta\gamma_i}, \forall i \in \{1, 2, ..., m\}$. $\mathcal{B}$ also calculates $g^{\alpha_i} = g^{\alpha\eta_i}, \forall i \in \{1, 2, ..., m\}$. Note that after this step, $\alpha_i = \alpha\eta_i$ and $\beta_i = \beta\gamma_i, \forall i \in \{1, 2, ..., m\}$. Now the adversary $\mathcal{B}$ sends $(g^{\alpha_1}, g^{\alpha_2}, ..., g^{\alpha_m}), (g^{\beta_1}, g^{\beta_2}, ..., g^{\beta_m})$ and $(\delta_1, \delta_2, ..., \delta_m)$ to $\mathcal{A}$. Now if $\rho = g^{\alpha\beta}$, then $(\delta_1, \delta_2, ..., \delta_m) = (\rho^{\gamma_1\eta_1}, \rho^{\gamma_2\eta_2}, ..., \rho^{\gamma_m\eta_m}) = (\{g^{\alpha\beta}\}^{\gamma_1\eta_1}, \{g^{\alpha\beta}\}^{\gamma_2\eta_2}, ..., \{g^{\alpha\beta}\}^{\gamma_m\eta_m}) = (\{g^{\alpha\eta_1}\}^{\beta\gamma_1}, \{g^{\alpha\eta_2}\}^{\beta\gamma_2}, ..., \{g^{\alpha\eta_m}\}^{\beta\gamma_m}) = (g^{\alpha_1\beta_1}, g^{\alpha_2\beta_2}, ..., g^{\alpha_m\beta_m})$; otherwise if $\rho = R$, then $(\delta_1, \delta_2, ..., \delta_m) = (R_1, R_2, ..., R_m)$. According to our assumption, upon receiving $(g^{\alpha_1}, g^{\alpha_2}, ..., g^{\alpha_m}), (g^{\beta_1}, g^{\beta_2}, ..., g^{\beta_m})$ and the challenge $(\delta_1, \delta_2, ..., \delta_m)$ from $\mathcal{B}$, $\mathcal{A}$ can distinguish between $(g^{\alpha_1\beta_1}, g^{\alpha_2\beta_2}, ..., g^{\alpha_m\beta_m})$ and $(R_1, R_2, ..., R_m)$. If the adversary $\mathcal{A}$ can distinguish between these two values, $\mathcal{B}$ can also distinguish between $g^{\alpha\beta}$ and R. Hence, $\mathcal{B}$ can identify the correct value of ρ. Thus, we have constructed an adversary $\mathcal{B}$ against the *assumption 1* (DDH assumption). Hence, the lemma 2 is proved. It can be easily verified that $Adv(\mathcal{A}) \leq Adv(\mathcal{B})$. □

Lemma 3: Given $g \in \mathbb{G}_q, \mathcal{G} = \{g^{x_i}, g^{y_i} : y_i = (\Sigma_{j=1}^{i-1} x_j - \Sigma_{j=i+1}^{\mu} x_j), i \in \{1, 2, ..., \mu\}\}$ and y_i are unknown for all $i \in \{1, 2, ..., \mu\}$. If $\Sigma_{i=1}^{\mu} v_i = \Sigma_{i=1}^{\mu} v_i'$ and $v_i, v_i' \in \mathbb{Z}_q$, then the bulletin boards A and B as given in Table I are indistinguishable.

$g^{y_1x_1}g^{v_1}$	$g^{y_1x_1}g^{v_1'}$
$g^{y_2x_2}g^{v_2}$	$g^{y_2x_2}g^{v_2'}$
...	...
...	...
...	...
$g^{y_\mu x_\mu}g^{v_\mu}$	$g^{y_\mu x_\mu}g^{v_\mu'}$
A	B

TABLE I
INDISTINGUISHABILITY OF BULLETIN BOARD A AND B

Proof: Without loss of generality, we assume that $\Sigma_{i=1}^{\mu} v_i = \Sigma_{i=1}^{\mu} v_i' = v$. Since $y_i = \Sigma_{j=1}^{i-1} x_j - \Sigma_{j=i+1}^{\mu} x_j$ for all $i \in \{1, 2, ..., \mu\}\}$, we have $\Sigma_{j=1}^{\mu} x_j y_j = 0$. Thus, we may write $g^{x_\mu y_\mu} = (1/g^{\Sigma_{i=1}^{(\mu-1)} x_i y_i})$. Also, since $\Sigma_{i=1}^{\mu} v_i = v$, we may write $g^{v_\mu} = (g^v / g^{\Sigma_{i=1}^{\mu-1} v_i})$. Note that y_i's are unknown for all $i \in \{1, 2, 3, .., \mu\}$. From Lemma 2, we can write $A = (g^{y_1x_1}g^{v_1}, g^{y_2x_2}g^{v_2}, \ldots, g^{y_{\mu-1}x_{\mu-1}}g^{v_{\mu-1}}, g^{y_\mu x_\mu}g^{v_\mu}) = (g^{y_1x_1}g^{v_1}, g^{y_2x_2}g^{v_2}, \ldots, g^{y_{\mu-1}x_{\mu-1}}g^{v_{\mu-1}}, g^v / \prod_{i=1}^{\mu-1} g^{y_i x_i} g^{v_i}) \approx_c (R_1 * g^{v_1}, R_2 * g^{v_2}, ..., R_{\mu-1} * g^{v_{\mu-1}}, g^v / \prod_{i=1}^{\mu-1} R_i * g^{v_i}) \approx_c (R_1, R_2, ..., R_{\mu-1}, g^v / \prod_{i=1}^{\mu-1} R_i) \approx_c (R_1 * g^{v_1'}, R_2 * g^{v_2'}, \ldots, R_{\mu-1} * g^{v_{\mu-1}'}, g^v / \prod_{i=1}^{\mu-1} R_i * g^{v_i'}) \approx_c (g^{y_1x_1}g^{v_1'}, g^{y_2x_2}g^{v_2'}, \ldots, g^{y_{\mu-1}x_{\mu-1}}g^{v_{\mu-1}'}, g^v / \prod_{i=1}^{\mu-1} g^{y_i x_i} g^{v_i'}) = (g^{y_1x_1}g^{v_1'}, g^{y_2x_2}g^{v_2'}, \ldots, g^{y_{\mu-1}x_{\mu-1}}g^{v_{\mu-1}'}, g^{y_\mu x_\mu}g^{v_\mu'}) = B$, since $g^{v_\mu'} = (g^v / g^{\Sigma_{i=1}^{\mu-1} v_i'})$. □

Lemma 4: Let us assume that $R = \{a_1, a_2, ..., a_k\}$, and $\nu = \Sigma_{j=1}^{k} a_j$. Given $g \in \mathbb{G}_q$, $X_i = (X_{i1}, X_{i2}, ..., X_{ik})$ and $Y_i = (Y_{i1}, Y_{i2}, ..., Y_{ik})$, where $X_{ij} = g^{x_{ij}}, Y_{ij} = g^{y_{ij}}, y_{ij} = (\Sigma_{l=1}^{i-1} x_{lj} - \Sigma_{l=i+1}^{\mu} x_{lj})$ and y_{ij} is unknown to the attacker $\forall j \in \{1, 2, ..., k\}, \forall i \in \{1, 2, ..., \mu\}, \mu \in \mathbb{N}$, the two bulletin boards A and B as in Table II and Table III respectively are indistinguishable, where

1. $v_{ij}, v'_{ij} \in R, \forall j \in \{1,2,..,k\}, \forall i \in \{1,2,...,\mu\}$
2. $\cup_{j=1}^{k} v_{ij} = \cup_{j=1}^{k} v'_{ij} = R, \forall i \in \{1,2,...,\mu\}$
3. $\Sigma_{j=1}^{k} v_{ij} = \Sigma_{j=1}^{k} v'_{ij} = \nu, \forall i \in \{1,2,...,\mu\}$
4. $\Sigma_{i=1}^{\mu} v_{ij} = \Sigma_{i=1}^{\mu} v'_{ij}, \forall j \in \{1,2,...,k\}$

$g^{y_{11}x_{11}}g^{v_{11}}$	$g^{y_{12}x_{12}}g^{v_{12}}$	...	$g^{y_{1k}x_{1k}}g^{v_{1k}}$
$g^{y_{21}x_{21}}g^{v_{21}}$	$g^{y_{22}x_{22}}g^{v_{22}}$	...	$g^{y_{2k}x_{2k}}g^{v_{2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{y_{\mu 1}x_{\mu 1}}g^{v_{\mu 1}}$	$g^{y_{\mu 2}x_{\mu 2}}g^{v_{\mu 2}}$	...	$g^{y_{\mu k}x_{\mu k}}g^{v_{\mu k}}$

TABLE II
BULLETIN BOARD A

$g^{y_{11}x_{11}}g^{v'_{11}}$	$g^{y_{12}x_{12}}g^{v'_{12}}$	...	$g^{y_{1k}x_{1k}}g^{v'_{1k}}$
$g^{y_{21}x_{21}}g^{v'_{21}}$	$g^{y_{22}x_{22}}g^{v'_{22}}$	...	$g^{y_{2k}x_{2k}}g^{v'_{2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{y_{\mu 1}x_{\mu 1}}g^{v'_{\mu 1}}$	$g^{y_{\mu 2}x_{\mu 2}}g^{v'_{\mu 2}}$	...	$g^{y_{\mu k}x_{\mu k}}g^{v'_{\mu k}}$

TABLE III
BULLETIN BOARD B

Proof: The proof follows from Lemma 3. □

Lemma 5: Let us assume that $R = \{a_1, a_2, ..., a_k\}$, and $\nu = \Sigma_{j=1}^{k} a_j$. Given $g \in \mathbb{G}_q$, $X_i = (X_{i1}, X_{i2}, ..., X_{ik})$ and $Y_i = (Y_{i1}, Y_{i2}, ..., Y_{ik})$, where $X_{ij} = g^{x_{ij}}, Y_{ij} = g^{y_{ij}}, y_{ij} = (\Sigma_{l=1}^{i-1} x_{lj} - \Sigma_{l=i+1}^{\mu} x_{lj})$ and y_{ij} is unknown $\forall j \in \{1,2,...,k\}, \forall i \in \{1,2,...,\mu\}, \mu \in \mathbb{N}$, the two bulletin boards A and B as in Table IV and V respectively are indistinguishable, where

1. $v_{ij}, v'_{ij} \in R, \forall j \in \{1,2,..,k\}, \forall i \in \{1,2,...,n-\mu\}$
2. $\cup_{j=1}^{k} v_{ij} = \cup_{j=1}^{k} v'_{ij} = R, \forall i \in \{1,2,...,n-\mu\}$
3. $\Sigma_{j=1}^{k} v_{ij} = \Sigma_{j=1}^{k} v'_{ij} = \nu, \forall i \in \{1,2,...,n-\mu\}$
4. $\Sigma_{i=1}^{n-\mu} v_{ij} = \Sigma_{i=1}^{n-\mu} v'_{ij} = v_j, \forall j \in \{1,2,...,k\}$

Proof: Let us choose votes $v_{ij} \in R$, for all $i \in \{n-\mu+1, n-\mu+2, ..., n\}$ and for all $j \in \{1,2,...,k\}$ such that $\cup_{j=1}^{k}\{v_{ij}\} = R$, for all $i \in \{n-\mu+1, n-\mu+2, ..., n\}$. Now we set $v'_{ij} = v_{ij}$, for all $i \in \{n-\mu+1, n-\mu+2, ..., n\}$ and for all $j \in \{1,2,...,k\}$. After this step, $\Sigma_{i=1}^{n} v_{ij} = \Sigma_{i=1}^{n} v'_{ij} = v_j + \Sigma_{i=n-\mu+1}^{n} v_{ij}$, for all $j \in \{1,2,...,k\}$. Now, if an attacker can distinguish between the two bulletin boards, she will be able to disprove Lemma 4. □

Lemma 6: Considering a partial collusion where an attacker colludes with $\mu < (n-1)$ voters, the attacker will only learn the partial tally of $n-\mu$ honest voters, not the individual votes of honest voters.

Proof: Without loss of generality, we assume that $\{V_{n-\mu+1}, V_{n-\mu+2}, ..., V_n\}$ is the set of colluding voters and $\{V_1, V_2, ..., V_{n-\mu}\}$ is the set of honest voters. In the proposed Borda count scheme, each voter V_i chooses $(x_{i1}, x_{i1}, ..., x_{ik})$ uniformly at random from $(\mathbb{Z}_q)^k$ and publishes $(g^{x_{i1}}, g^{x_{i2}}, ..., g^{x_{ik}})$ in the first round along with k zero-knowledge proofs $P_K\{x_{ij}\}, \forall j \in \{1,2,...,k\}$. In the second round, each voter V_i publishes $(Z_{i1}, Z_{i2}, ..., Z_{ik})$, where $Z_{ij} = \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}}, y_{ij} = \Sigma_{l<i} x_{lj} - \Sigma_{l>i} x_{lj}, \forall j \in \{1,2,...,k\}$. The attacker will know votes of the colluding voters and their randomness, i.e. attacker will know $(v_{i1}, v_{i2}, ..., v_{ik})$ and $(x_{i1}, x_{i1}, ..., x_{ik})$ for each colluding voter V_i, where $i \in \{n-\mu+1, n-\mu+2, ..., n\}$. Therefore, the attacker can compute $(Z_{i1}, Z_{i2}, ..., Z_{ik})$ for each colluding voter V_i, where $i \in \{n-\mu+1, n-\mu+2, ..., n\}$. According to the protocol, $y_{ij} = (\Sigma_{l=1}^{i-1} x_{lj} - \Sigma_{l=i+1}^{n} x_{lj}), \forall j \in \{1,2,...,k\}, \forall i \in \{1,2,...,n\}$. From Lemma 1, in case of partial collusion, we can conclude that y_{ij}'s are unknown $\forall j \in \{1,2,...,k\}, \forall i \in \{1,2,...,n\}$. Hence. the attacker's view of the bulletin board will be same as given in Table IV.

Now according to the Lemma 5, the attacker will not be able to distinguish the two bulletin boards given in Table IV and Table V with the same partial tally for honest voters. Therefore, the attacker will only learn the partial tally of honest voters, not the individual votes of honest voters.

$g^{u_{11}}g^{v_{11}}$	$g^{u_{12}}g^{v_{12}}$	...	$g^{u_{1k}}g^{v_{1k}}$
$g^{u_{21}}g^{v_{21}}$	$g^{u_{22}}g^{v_{22}}$	...	$g^{u_{2k}}g^{v_{2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{u_{n-\mu 1}}g^{v_{n-\mu 1}}$	$g^{u_{n-\mu 2}}g^{v_{n-\mu 2}}$	...	$g^{u_{n-\mu k}}g^{v_{n-\mu k}}$
$g^{u_{n-\mu+11}}$	$g^{u_{n-\mu+12}}$	...	$g^{u_{n-\mu+1k}}$
$g^{u_{n-\mu+21}}$	$g^{u_{n-\mu+22}}$	...	$g^{u_{n-\mu+2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{u_{n1}}$	$g^{u_{n2}}$	...	$g^{u_{nk}}$

TABLE IV
BULLETIN BOARD A, WHERE $u_{ij} = y_{ij}x_{ij}, \forall j \in \{1,2,...,k\}$ AND $\forall i \in \{1,2,...,n\}$.

$g^{u_{11}}g^{v'_{11}}$	$g^{u_{12}}g^{v'_{12}}$	...	$g^{u_{1k}}g^{v'_{1k}}$
$g^{u_{21}}g^{v'_{21}}$	$g^{u_{22}}g^{v'_{22}}$	...	$g^{u_{2k}}g^{v'_{2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{u_{n-\mu 1}}g^{v'_{n-\mu 1}}$	$g^{u_{n-\mu 2}}g^{v'_{n-\mu 2}}$	...	$g^{u_{n-\mu k}}g^{v'_{n-\mu k}}$
$g^{u_{n-\mu+11}}$	$g^{u_{n-\mu+12}}$	...	$g^{u_{n-\mu+1k}}$
$g^{u_{n-\mu+21}}$	$g^{u_{n-\mu+22}}$	...	$g^{u_{n-\mu+2k}}$
...	...	...	...
...	...	...	...
...	...	...	...
$g^{u_{n1}}$	$g^{u_{n2}}$	...	$g^{u_{nk}}$

TABLE V
BULLETIN BOARD B, WHERE $u_{ij} = y_{ij}x_{ij}, \forall j \in \{1,2,...,k\}$ AND $\forall i \in \{1,2,...,n\}$.

B. Self-tallying

In our protocol, during the tallying phase, the final tally can be computed by anyone, including voters and observers of the protocol, without any external help. This can be easily checked by observing the protocol and the *Proposition 1*. In the first round of the protocol, voters choose their private key, and in the second round their public keys are combined in such a way that the random factors vanishes after the second round. Thus, during the tallying phase, the final tally can be computed correctly by any observer of the protocol. Therefore, our protocol satisfies the self-tallying property. The use of zero-knowledge proofs in the protocol is to ensure that all voters follow the protocol faithfully.

C. Dispute-freeness

Our protocol also satisfies the dispute-freeness property. The use of k non-interactive zero-knowledge proofs in the first round of the protocol ensures that the voter knows the secret keys $(x_{i1}, x_{i2}, ..., x_{ik})$ corresponding to the public keys $(g^{x_{i1}}, g^{x_{i2}}, ..., g^{x_{ik}})$. The use of k non-interactive Cramer, Damgård, and Schoenmakers [17] zero-knowledge proofs ensures that each ballot will encode exactly one permutation of $(a_1, a_2, ..., a_k)$. Furthermore, since we use a public authenticated channel, any attempt to cast more than one vote can be detected by other participants of the protocol. Thus, our protocol enforces the one-man-one-vote requirement, which, combined with the public verifiablity of all operations in the protocol, ensures dispute-freeness.

D. Limitation

In this section, we discuss the limitations of our protocol. One limitation is that all voters must follow the protocol till the tally process. If some voters do not send data in the second round of the protocol, no one will be able to compute the tally, and hence the tallying will fail. However, this attack is conspicuous since everyone will be able to identify the attackers. In that case, voters can exclude the attackers from the election and restart the protocol to recover from the attack. Note that the voter's privacy is still preserved. Nevertheless, there would be delay in the election process.

In our implementation of the protocol using Ethereum Blockchain and smart contract, we use a deposit and refund paradigm [40] to enforce a financial incentive to all voters as a countermeasure to this issue. All voters are required to deposit some money into the smart contract to register for the election. This deposited money is refunded to the voter after she successfully casts her ballot. However, a registered voter will lose her deposit if she does not cast her vote successfully in the second round of the protocol.

Our protocol is not coercion-resistant since a voter can be coerced to vote for a particular candidate and to reveal her secret parameters to prove how she voted. This is another limitation of our system. Normally, this kind of coercion resistance is provided in a supervised environment like in a polling station [35]. However, in decentralized environment where the voting process is unsupervised, providing coercion resistance seems difficult.

Here, we mention that the above limitations also exist in the Kiayias and Yung [38], Groth [29] and open vote network [30] protocols. Although a centralised voting protocol that is executed in a supervised environment with trusted election authorities may prevent this kind of issues, however, the trustworthiness of the election authorities is often called into question. In our proposed protocol, the tally can be computed by voters themselves (and any observer of the protocol) without involving any tallying authority.

Therefore, for a small-scale election where the above limitations are not of great concern, the proposed decentralised Borda count protocol can prove to be useful and efficient.

VI. Performance analysis of the protocol

In this section, we analyze the computation and communication cost of our proposed protocol. There are only two rounds in our protocol. As discussed in [27], in any secure multy-party computation protocol, minimum two rounds are required to compute a function securely. Since exponentioations are the most expensive operations in our protocol, we analyze our protocol in terms of the number of exponentiations.

We assume that the number of candidates competing in the election is k. In the first round, each voter needs to do k exponentiations to generate public keys. In addition, there are k NIZK proofs corresponding to their k secret keys. Each of these NIZK proofs requires one exponentiation for creation and 1.2 exponentiations for verification. (Here we do not count the cost of validating a public key, which requires one exponentiation in the finite field setting but is essentially free in the elliptic curve setting.) We assume that the simultaneous multiple exponentiation (SME) technique [47] is used to optimize computations. Using the SME method, a term of the form $g^x h^y$ requires about 1.2 exponentiations to calculate. Hence, a voter needs to do $2k$ exponentiations in the first round to create her public keys along with NIZK proofs. In order to verify these NIZK proofs, one needs to do $1.2k$ exponentiations. In the second round, a voter needs to do k exponentiations to generate the ballot. In addition, the voter needs to create k 1-out-of-k NIZK proofs. Each of these 1-out-of-k NIZK proofs requires $(2.4(k-1)+2)$ exponentiations for generation and $2.4k$ exponentiations for verification of the same. Hence, all k 1-out-of-k NIZK proofs require $(2.4k^2 - 0.4k)$ exponentiations for generation and $2.4k^2$ for verification. Hence, in the second round, a voter needs to do $(k + ((2.4k^2 - 0.4k)))$ exponentiations to generate her ballot. All together, a voter needs to do $(2k + (k + ((2.4k^2 - 0.4k))))$ exponentiations i.e. $(2.4k^2 + 2.6k)$ exponentiations to cast her vote in the election. The total number of exponentiations required to verify a ballot is equal to $(1.2k + 2.4k^2)$. In the first round of the protocol, the size of public keys of a voter is k elements of group the $\mathbb{G}_q$. In addition, in the first round, the size of k NIZK proofs is k elements of $\mathbb{Z}_q$ plus k elements of $\mathbb{G}_q$. Hence, the size of the public keys along with NIZK proofs is $2k$ elements of the group $\mathbb{G}_q$ plus k elements of the group $\mathbb{Z}_q$. In the second round of the protocol, the size of each ballot is k elements of the group $\mathbb{G}_q$. In addition, the size of each of k 1-out-of-k NIZK proofs is $2k$ elements of $\mathbb{Z}_q$ plus $2k$ elements of $\mathbb{G}_q$. Hence, in the second round, the size of each ballot along with k 1-out-of-k NIZK proofs is $(k + 2k^2)$ elements of the group $\mathbb{G}_q$ plus $2k^2$ elements of the group $\mathbb{Z}_q$. Therefore, all together, the total space required by a voter is $(3k + 2k^2)$ elements of the group $\mathbb{G}_q$ plus $(2k^2 + k)$ elements of the group $\mathbb{Z}_q$.

Table VI and Table VII highlight the computation and communication cost (space) respectively for the proposed Borda count protocol when k candidates compete in the election.

Note that the OV-Net protocol requires $(2.4k - 0.4)$ exponentiations (using the SME technique) for generation and $2.4k$ exponentiations for verification of the NIZK proof in the second round of the protocol. Therefore, the number

of exponentiations required in the OV-Net protocol is $O(k)$ that is linear with the number of candidates contesting in the election. However, our proposed Borda count protocol requires $(2.4k^2 - 0.4k)$ exponentiations for generation and $2.4k^2$ exponentiations for verification of the NIZK proofs in the second round of the protocol. Therefore, the number of exponentiations required in our proposed protocol is $O(k^2)$ that is quadratic with the number of candidates competing in the election. Our protocol requires significantly more computation than OV-net which is based on plurality voting, because a ranked-choice voting system is inherently more complex than a non-ranking based voting system. However, a ranked-choice voting system tends to give a fairer outcome as the system takes in more information from voters about their preferences than a non-ranking based system (e.g., plurality).

First round			Second round		
Public keys	NIZKP	Total	Ballot	NIZKP	Total
k	k	$2k$	k	$2.4k^2-0.4k$	$2.4k^2+0.6k$

TABLE VI
THE COMPUTATION COST FOR THE PROPOSED SCHEME IN NUMBER OF EXPONENTIATIONS WHEN k CANDIDATES COMPETE IN THE ELECTION.

First round			Second round		
Public keys	NIZKP	Total	Ballot	NIZKP	Total
$k \cdot a$	$k \cdot a + k \cdot b$	$2k \cdot a + k \cdot b$	$k \cdot a$	$2k^2 \cdot a + 2k^2 \cdot b$	$(k+2k^2) \cdot a + 2k^2 \cdot b$

TABLE VII
THE COMMUNICATION COST (SPACE) FOR THE PROPOSED SCHEME WHEN k CANDIDATES COMPETE IN THE ELECTION. a AND b REPRESENT THE SIZE OF EACH ELEMENT OF THE GROUP $\mathbb{G}_q$ AND $\mathbb{Z}_q$ RESPECTIVELY.

We now compare the performance of our proposed Borda count ranked-choice voting system with some of the well-known non-ranking based voting systems. Several cryptographic voting protocols are proposed in the literature, however, most of them offer security and integrity assurance by introducing a set of trustworthy tallying authorities [38], [29]. For the purpose of comparison, we consider only the self-tallying voting protocols without involving any tallying authorities. Therefore, we compare our protocol mainly with the Kiayias-Yung [38], Groth [29] and OV-Net [30] protocols. Table VIII highlights a comparison between our proposed Borda count protocol and these three previously proposed plurality voting solutions.

In [38], Kiayias-Yung proposed a 3-round veto protocol. In the first round, each voter i chooses a random value x_i from $\mathbb{Z}_q$ and publishes her public key $P_i = g^{x_i}$. In this round, the voter needs to compute one exponentiation and the corresponding NIZK proof for the knowledge of the exponent. In the second round, each voter chooses n-vector private keys $(y_1, y_2, ..., y_n)$ uniformly at random from $\mathbb{Z}_q^n$ and publishes the corresponding public keys $(g^{y_1}, g^{y_2}, ..., g^{y_n})$, where n is the number of voters. To perform this step, a voter needs to compute n exponentiations. Each voter also computes $(P_1^{y_1}, P_2^{y_2}, ..., P_n^{y_n})$, which requires n more exponentiations. In round three, each voter performs one more exponentiation. After this step, the tally can be computed universally. Therefore, each voter needs to perform $(2n+2)$ exponentiations in total. For each exponentiation, the voter needs to publish the corresponding NIZK proof. Each voter publishes $O(nk)$ data, where k is the number of candidates contesting in the election, and n is the total number of voters. The final tally is computed from $O(n^2k)$ data. Table VIII summarizes the performance of this protocol.

Groth [29] investigated the Kiayias-Yung's protocol in order to reduce its system complexity. Groth's protocol has $(n+1)$ rounds, where n is the total number of voters. In round 1, each voter i publishes her public key. After this step, each voter sends her encrypted vote one after another depending on the result sent by the previous voter. As a result, instead of finishing the protocol in three rounds as in [38], the protocol requires $n+1$ rounds, where n is the total number of voters. In total, the first voter needs to compute three exponentiations and the corresponding NIZK proofs. All other voters need to compute four exponentiations along with four corresponding NIZK proofs in total. Each voter publishes $O(k)$ data, where k is the number of candidates contesting in the election. The final tally is computed from $O(nk)$ data.

The OV-Net protocol [30] executes in two rounds, which is more efficient than the Kiayias-Yung [38] and Groth's [29] protocols. In the first round, every voter publishes one public key along with a NIZK proof of the corresponding secret key. A voter needs to compute one exponentiation and one NIZK proof for the knowledge of the exponent. In the second round, each voter sends her encrypted vote along with one 1-out-of-k NIZK proof. In this round, the voter needs to compute one exponentiation and one 1-out-of-k NIZK proof. Each voter publishes $O(k)$ data, k being the number of candidates contesting in the election. The final tally is computed from $O(nk)$ data, where n is the total number of voters.

All of the protocols discussed above are designed to implement a plurality voting electoral system. We propose a Borda count voting protocol that is a ranked-choice voting system. Our proposed protocol runs in two rounds. In the first round, each voter V_i calculates $X_{i1} = g^{x_{i1}}, X_{i2} = g^{x_{i2}}, ..., X_{ik} = g^{x_{ik}}$ and publishes $(X_{i1}, P_K\{x_{i1}\}, X_{i2}, P_K\{x_{i2}\}, \ldots, X_{ik}, P_K\{x_{ik}\})$, where $P_K\{x_{ij}\}$ is the NIZK proof for $x_{ij}; \forall j \in \{1, 2, ..., k\}$. A voter needs to compute k exponentiations and k NIZK proofs for the exponents. In the second round, each voter V_i publishes $(Z_{i1}, Z_{i2}, ..., Z_{ik})$ and k 1-out-of-k NIZK proofs $\Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\}$, where $Z_{ij} = \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}}; \forall j \in \{1, 2, ..., k\}$. Our protocol adopts the same cancelling technique as in [30] to achieve self-tallying without involving any tallying authority, but our system is designed to support ranked-choice voting instead of plurality voting. The construction and verification of zero-knowledge proofs are different from [30]. In our system, each voter publishes $O(k^2)$ data, where k is the number of candidates contesting in the election. The final tally is computed from $O(nk^2)$ data, where n is the total number of voters.

Protocols	Election type	Rounds	Exp	NIZKP for exponent	NIZKP for equality	1-out-of-k NIZKP	Total traffic	Total computation
Kiayias-Yung [38]	Plurality (Non-ranking based)	3	$2n+2$	$n+1$	n	1	$O(n^2k)$	$O(n^2k)$
Groth [29]	Plurality (Non-ranking based)	$n+1$	4	2	1	1	$O(nk)$	$O(nk)$
OV-Net [30]	Plurality (Non-ranking based)	2	2	1	0	1	$O(nk)$	$O(nk)$
Proposed protocol	Borda count (rank-choice based)	2	$2k$	k	0	k	$O(nk^2)$	$O(nk^2)$

TABLE VIII
COMPARISON WITH RELATED PROTOCOLS PROPOSED IN THE LITERATURE. THE NUMBER OF PARTICIPANTS IN THE ELECTION IS n, AND THE NUMBER CANDIDATES CONTESTING IN THE ELECTION IS k.

VII. A SMART CONTRACT IMPLEMENTATION

The proposed decentralized Borda Count protocol is suitable for implementation over a Blockchain. In this paper, we propose a smart contract implementation of our protocol on Ethereum in order to enforce the execution of the voting protocol. We are using Ethereum since it can store and execute programs that are written as smart contracts. Ethereum's consensus mechanism enforces the correct execution of these smart contracts. Its peer-to-peer network serves as an authenticated public channel. We use the blockchain as a public bulletin board as well as to enforce the execution of the election process in a timely manner.

A. Ethereum

An Ethereum transaction is a digitally signed instruction constructed by a user of the Ethereum blockchain. There are two types of transactions: those that create smart contracts and those that are responsible for message calls. An Ethereum transaction consists of several fields, which specify the sender's and the receiver's addresses, and the transaction data [46].

There are two types of accounts available in Ethereum blockchain. They are called Contract Account and Externally Owned Account.

A user creates a smart contract using her externally owned account. The Ethereum currency 'Ether' can be stored in both of these types of accounts. A user needs to purchase 'gas' using 'Ether' currency in order to execute a smart contract. The gas price is set according to the conversion rate of ether to gas. For each assembly 'opcode' (instruction), there is a fixed gas cost to execute the instruction depending on its execution time. The cost of gas to execute a transaction is regarded as the transaction fee. This transaction fee is an incentive for the miners to add the transaction into their block.

The integrity of the Ethereum blockchain depends on its 'proof-of-work' mechanism. The 'proof-of-work' is a computationally difficult puzzle that must be solved by a miner in order to get a block added to the blockchain. Ethereum's blockchain is a simplified version of the GHOST protocol introduced by Sompolinsky and Zohar [59]. In Ethereum's blockchain, blocks are created in every 12 seconds interval. As a result, there is a possibility that two or more blocks would be created at the same time by different miners, and hence some blocks would be discarded. These discarded blocks are added to the blockchain as 'uncle blocks', and the corresponding miners still get some rewards in 'Ether' currency. Furthermore, if the same smart contract is called by multiple transactions, the final state of the smart contract is determined by the order of execution of the transactions.

We now discuss our implementation of the voting protocol on Ethereum in the following sections.

B. Structure of Implementation

We follow the similar design architecture as proposed by McCorry et al. in [46]. However, in [46], McCorry et al. implemented the OV-Net protocol only for a two-candidate election ('yes'/'no' voting). We implement the proposed Borda count protocol for multiple candidates who are contesting in the election. Note that OV-Net protocol is designed for plurality voting which is a non-ranking based voting system. However, our proposed protocol is designed for Borda count voting which is a 'ranking-based' voting system. The two election processes are different. In addition, the structure of ballots and the NIZK proofs are also different.

While implementing our protocol over Ethereum, some of the tasks exceeded the maximum gas limit (approximately 8 million gas as in June, 2019) of a block. In [46], a single transaction is sufficient to perform a task, however, due to the complexity of the Borda count system, we had to send multiple transactions in parallel to complete some tasks. We split the tasks into several smaller tasks (within the gas limit) in order to execute them in parallel. We have developed two smart contracts in Solidity language:

(1) **VoteContract:** The voting protocol is implemented in this contract. This contract also controls the voting process and checks the validity of all zero-knowledge proofs.

(2) **CryptoContract:** This contract consists of the code for creating zero-knowledge proofs. All voters call this smart contract to create zero-knowledge proofs so that the same code can be used by all voters. This contract can be executed locally without interacting with Ethereum blockchain.

The election administrator is the owner of these two smart contracts. We have also developed the following three HTML5/Javascript pages to provide browser interfaces for the users:

(1) **administrator.html:** This page is constructed for the administrator to manage the entire election process. The election administrator sets a list of eligible voters and a list of timers using this page. The timers are used to set the deadlines for various stages of the election process so the voting process progresses in a timely manner. This page also contains the code to notify the Ethereum blockchain to begin the registration process, to close the registration process and

begin to cast the vote, to close the voting and compute the tally based on these timers.

(2) **voter.html:** This page is implemented to facilitate the voter to register and cast her vote.

(3) **live.html:** This page is dedicated for any observer of the election process to watch the progress of the election including the beginning and ending of each stage, voter registration process and vote casting process. This page shows the addresses of current registered voters, addresses of the voters who have committed their vote and addresses of the voters who have already cast their vote. However, it is not possible to compute the running tally.

In addition, we have implemented a Java program to locally generate the random private keys $(x_{i1}, x_{i2}, ..., x_{ik})$ and the corresponding public keys $(g^{x_{i1}}, g^{x_{i2}}, ..., g^{x_{ik}})$ for a voter. The private keys are kept secret by the voter. These keys are required by the voter's web browser voter.html to register and cast her vote.

We use the Web3 framework provided by the Ethereum foundation for communication between the user's web browser and the Ethereum client. In our implementation, the user does not need to directly interact with the Ethereum client, and it is sufficient if the Ethereum client runs in the background. The election administrator and all voters need to have their own Ethereum account. The election administration sets the list of eligible voters using their Ethereum account address. Users (including voters and the election administrator) need to unlock their account using their password to send transactions to the blockchain from their web browser. The password is used to decrypt their private key that is necessary to generate the digital signature of the transaction.

C. Election Stages

The election runs in five stages in our implementation. Figure 1 depicts these stages. The election administrator arranges the election process by providing a list of timers and eligible voters to the smart contract. The contract only permits eligible voters to register and cast their vote before the deadline as specified by the list of timers. In addition, as mentioned before, an eligible voter may need to deposit ether in the contract to register for the election. This deposit is automatically refunded to the voter after her vote is successfully accepted by Ethereum blockchain. We now describe these election stages in detail.

Setup: At this stage, the election administrator sets a list of eligible voters using their Ethereum account address, a list of timers and the registration deposit d on the Ethereum blockchain. The election administrator also sets whether the optional commit stage should be enabled or not. The timers are used to set the closing time (deadline) for each stage of the election to enforce that the election process progresses in a predefined timely manner. The list of timers includes the following timers.

$-T_{endreg}$*:* Voters must register for the election before this time limit. To register for the election, a voter needs to send her voting keys along with the NIZK proofs $(g^{x_{i1}}, P_K\{x_{i1}\}, g^{x_{i2}}, P_K\{x_{i2}\}, ..., g^{x_{ik}}, P_K\{x_{ik}\})$, where $P_K\{x_{ij}\}$ is the NIZK proof of $x_{ij}; \forall j \in \{1, 2, ..., k\}$ and k is the number of candidates competing in the election, to the blockchain before this time.

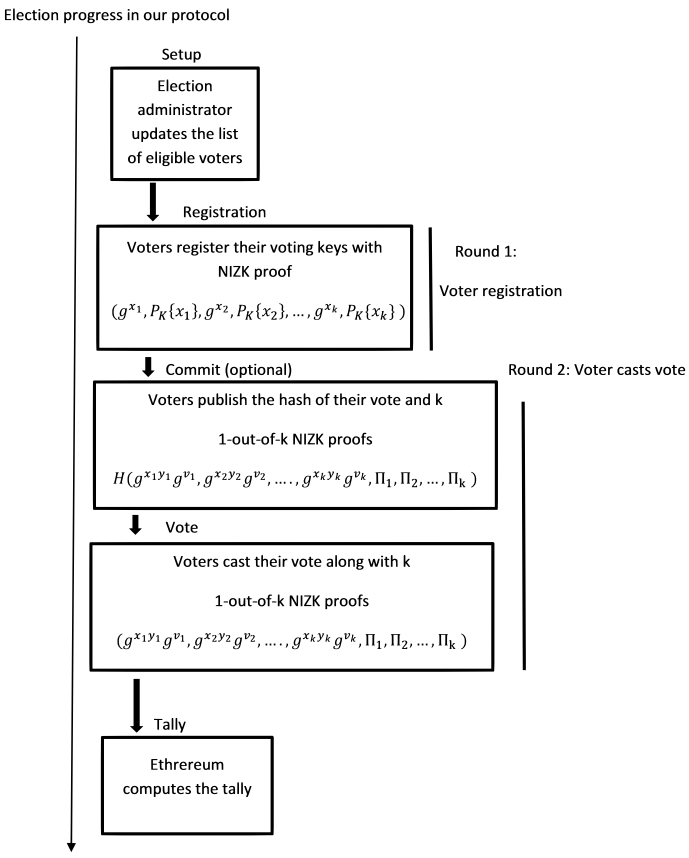

Fig. 1. Election stages in our protocol implementation.

$-T_{startvote}$*:* This represents the time before which the blockchain must be notified by the election administrator to start the election (i.e. the second round of our protocol). If the election administrator fails to notify the blockchain by this time, the voting process will be aborted, and all registered voters will get their deposit back.

$-T_{endcommit}$*:* If the Commit (optional) stage is enabled by the election administrator in this Setup stage, voters must send the hash of their vote to the blockchain as a commitment before this time. All voters send $H((Z_{i1}, Z_{i2}, ..., Z_{ik}), \Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\})$, where $\Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\}$ are k non-interactive zero-knowledge proofs corresponding to the vote in the second round of our protocol and $Z_{ij} = \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}}; \forall j \in \{1, 2, ..., k\}$, k is the number of candidates competing in the election. This time is effective only if the Commit stage (optional) is enabled. If some registered voters fail to commit their vote by this time, the election process will be aborted. In that case, all other registered voters who have successfully committed their vote by this time can get their deposit back.

$-T_{endvote}$*:* This represents the time before which all voters must cast their vote in the election. All voters send their vote $((Z_{i1}, Z_{i2}, ..., Z_{ik}), \Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\})$ as defined in the second round of the protocol. If some registered voters fail to cast their vote by this time, the

election process will be aborted. The deposit of those voters will be confiscated. However, all other registered voters who have successfully cast their vote by this time will automatically get their deposit back.

$-T_{min}$*:* This represents the minimum length of time in which the commitment stage and voting stage must remain active. In a decentralized voting protocol, voters should be given sufficient time to commit and cast their votes. Therefore, the commitment and voting stage should remain active for a sufficient amount of time.

At the end of this stage, the election administrator notifies the Ethereum blockchain to change the state (of the contract) from the Setup to the Registration stage.

Registration: At this stage, all eligible voters first review the parameters of the election set by the election administrator such as timers (closing time for each stage of the election) and the registration deposit d. Then eligible voters may choose to register for the election. Let there be k candidates competing in the election. To register for the election, a voter needs to compute her voting keys along with the non-interactive zero-knowledge proofs. The voter sends the voting keys along with NIZK (non-interactive zero-knowledge) proofs to the blockchain along with a deposit of d ether. Ethereum verifies the NIZK proofs by executing the smart contract code and adds the transaction to the blockchain upon successful verification of the NIZK proofs. Ethereum rejects all registration request transactions that are received after T_{endreg}. At the end of this stage, the election administrator notifies the blockchain to change the state from Registration to either the optional Commit stage or the Vote stage depending on whether the optional Commit stage is enabled or not. During this transition, Ethereum computes all voter's reconstructed keys $(g^{y_{i1}}, g^{y_{i2}}, ..., g^{y_{ik}})$, where $g^{y_{ij}} = \prod_{l=1}^{i-1} g^{x_{lj}} / \prod_{l=i+1}^{n} g^{x_{lj}}; \forall j \in \{1, 2, ..., k\}$.

Commit (optional): If the optional commit stage is enabled, all registered voters publish the hash of their vote and NIZK proofs $H((Z_{i1}, Z_{i2}, ..., Z_{ik}), \Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\})$ to the blockchain. Ethereum rejects all commitment transactions that are received after $T_{endcommit}$. After receiving commitments from all registered voters, the contract automatically changes its state to the Vote stage.

Vote: All voters cast their vote at this stage. All voters publish their ballot - $(Z_{i1}, Z_{i2}, ..., Z_{ik}), \Pi_j[x_{i1}, x_{i2}, ..., x_{ik} : X_i, Z_i]; \forall j \in \{1, 2, ..., k\}$ to the blockchain. Ethereum accepts the transaction if the verification of all the NIZK proofs succeeds. The smart contract refunds the deposit d to the voter after successfully accepting her ballot (encrypted vote). Ethereum rejects all ballots that are received after $T_{endvote}$. The administrator notifies Ethereum to change the state of the contract to the Tally stage after all voters cast their ballots.

Tally: At this stage, Ethereum computes the tally for each candidate once it receives the corresponding notification from the election administrator. To compute the tally for the j-th candidate, Ethereum evaluates $\prod_{i=1}^{n} \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}} = g^{\Sigma_{i=1}^{n} v_{ij}}$, where $j \in \{1, 2, ..., k\}$. Ethereum then computes the total score $\Sigma_{i=1}^{n} v_{ij}$ obtained by the j-th candidate using brute force search, for each $j \in \{1, 2, ..., k\}$. Note that this brute force search is feasible since the tally is a small number.

D. Design choices

We now focus on the design choices for implementing our protocol on Ethereum.

Score associated with each rank and the tally computation. In a Borda count protocol, a score is associated with each rank. Let there be k candidates contesting in the election. In our proposed scheme, a voter's vote will be of the form $(v_1, v_2, ..., v_k)$, where $(v_1, v_2, ..., v_k)$ is a permutation of $(a_1, a_2, ..., a_k)$. The score a_j is associated with the j-th rank and $a_{j-1} > a_j, \forall j \in \{1, 2, .., k\}$ i.e a higher score implies higher rank of the candidate. In our implementation, we set the following scores associated to each rank: $a_1 = k, a_2 = k-1, ..., a_k = 1$. Therefore, at the tallying phase, the total score obtained by a candidate (final tally) can be at most nk, where n is the number of participants in the election. Furthermore, the minimum total score that can be obtained by a candidate is n. In our proposed protocol, at the tallying stage, anyone can compute $\prod_{i=1}^{n} \{g^{y_{ij}}\}^{x_{ij}} g^{v_{ij}} = g^{\Sigma_{i=1}^{n} v_{ij}}$, where $j \in \{1, 2, ..., k\}$. The total score obtained by the j-th candidate is $\Sigma_{i=1}^{n} v_{ij}$. Therefore, $n \leq \Sigma_{i=1}^{n} v_{ij} \leq nk$, which is normally a small number. Thus, the exhaustive search (brute-force) requires at most $(n(k-1)+1)$ operations to compute the discrete logarithm of $g^{\Sigma_{i=1}^{n} v_{ij}}$ for all $j \in \{1, 2, .., k\}$, where n is the number of participants in the election.

The election administrator. In our implementation, the election administrator sets the parameters of the election, list of eligible voters. The election administrator notifies the smart contract to begin the Registration stage. Note that a smart contract can not start executing code without user interaction. Therefore, there must be a user who can take the responsibility of notifying the smart contract to begin the election process and compute the tally. We assume that the election administrator notifies the blockchain. A registered voter can also notify the blockchain.

Preventing re-entrancy attack. This attack is possible if a smart contract sends 'ether' to a user before deducing their balance. An attacker can exploit this vulnerability by calling the smart contract recursively in such a way that sending of ether is repeated, however, the deduction of balance is done only once. Using re-entrancy vulnerability in DAO, an attacker stole 3.6 million ether. Luu et al. [45] analyze that there are 186 distinct smart contracts stored on the blockhain vulnerable to this attack. To thwart this attack, Reitwiessner [52] suggests to first deduce the balance before sending the ether. We follow the same in our implementation.

Preventing replay attack. In the first round of our protocol, all voters send publicly their public keys with NIZK proofs $(g^{x_{i1}}, P_K\{x_{i1}\}, g^{x_{i2}}, P_K\{x_{i2}\}, ..., g^{x_{ik}}, P_K\{x_{ik}\})$. Another eligible voter might replay the same keys and NIZK proofs but without actually knowing the corresponding private keys. This will invalidate the purpose of the NIZK proof which is supposed to enforce the user to prove the knowledge of the private key. To prevent this replay attack, we follow the technique suggested in [30], by including the sender's unique identity (*msg.sender*) as an input argument to the hash

function in the NIZK proofs. Therefore, no other voter can use the same NIZK proof as every sender's identity over the blockchain is different.

Downloading full blockchain. Currently, it is necessary to download the full Ethereum blockchain to verify the correct execution of the voting protocol. Note that a voter can participate in the election without downloading the full Ethereum blockchain, however in that case, the voter needs to trust Ethereum blockchain for correctly executing the protocol. The voter only needs to broadcast her transactions in the Ethereum blockchain network. They can use live.html page for confirmation of their registration and to view that their vote has been included in the blockchain. This enables voters with limited resources to cast their vote.

Refund of the deposit. In our implementation, all registered voters must send their ballots before the deadline so as to ensure that their ballots are included in the blockchain. The deposit d acts as a financial incentive that is refunded to the voters who adhere to the protocol and submit their ballots in time. However, those voters who fail to do so in time will face forfeiture of their deposit. The confiscated deposit can be used to compensate compliant votes or to support a charitable cause. The deposit is automatically refunded to a voter if either the voting process is successfully completed or if the voting process is aborted due to some other voter's failure in adhering to the protocol.

E. Experiment on Ethereum

We deployed our implementation on Ethereum's private network that mimics the production network. We have tested our implementation for different numbers of voters and for different candidates competing in the election. We present the results of our experiments with eighty voters and a varying number of candidates. In Table IX, we have outlined each transaction's computational and financial cost for eighty voters and five candidates contesting the election.

The total number of transactions in an election depends on the number of voters and the number of candidates competing in the election. The number of transactions that we sent to simulate the election process is shown in Table IX. The election administrator is denoted by the prefix 'AD:' and each voter is denoted by the prefix 'VT:' (see Table IX). Each transaction is a broadcast transaction. We have calculated the cost in US Dollar (denoted by '$' in the table) and rounded it to two decimal places.

Currently, the maximum gas limit of a single Ethereum block is approximately 8 million. The amount of gas required to store our code as a smart contract exceeds the gas limit of a single Ethereum block. Therefore, we had to create two smart contracts, namely, VoteContract and CryptoContract . The main protocol logic is implemented in the contract 'VoteContract', whereas, the code for the creation of the NIZK proofs is implemented in the contract 'CryptoContract'. The verification of the NIZK proofs is implemented in the contract 'VoteContract'.

As shown in Table IX, the voter registration costs approximately 48% of the block capacity when five candidates are competing in an election. Note that, as mentioned before, the maximum block capacity in Ethereum blockchain is about 8 million gas as in June, 2019. Thus, the current block supports at most two voter registrations per block. The vote casting costs more than the capacity of a block. Therefore, we made five transactions to cast a vote. The reason is that the number of exponentiations increases with the number of candidates completing in the election, and hence the cost of verifying NIZK proofs also increases. As shown in Table IX, we have made five transactions to cast a vote when the number of candidates competing in the election is five. Each transaction of vote casting costs approximately 90% of the block capacity. This suggests that five blocks are needed to cast a vote. Note that, in Ethereum, currently blocks are generated at a rate of one block in every 12 seconds.

Entity: Transaction	A	B	C	D
AD: VoteContract	1	5,498,780	5,498,780	1.70
AD: CryptoContract	1	3,698,878	3,698,878	1.14
AD: Eligible	1	4,349,336	4,349,336	1.34
AD: Begin setup	1	257,209	257,209	0.08
VT: Register	1	3,850,910	3,850,910	1.19
AD: Begin election	5	7,149,237	35,746,186	11.05
VT: Commit	1	183,457	183,457	0.06
VT: Vote	5	7,213,571	36,067,855	11.14
AD: Tally	5	5,629,272	28,146,362	8.70
Administrator Total	14		77,696,751	24.01
Voter total	7		40,102,222	12.39

TABLE IX
A BREAKDOWN OF THE COST FOR EIGHTY PARTICIPANTS USING OUR PROTOCOL WITH FIVE CANDIDATES COMPETING IN THE ELECTION. THE COSTS ARE APPROXIMATED IN USD ('$') USING THE CONVERSION RATE OF 1 ETHER=$309 AND THE GAS PRICE OF 0.000000001 ETHER THAT ARE REAL WORLD COSTS IN JUNE, 2019. WE HAVE APPROXIMATED THE COST FOR THE ELECTION ADMINISTRATOR 'AD' AND THE VOTER 'VT'. COLUMNS ARE REPRESENTED AS - A: NUMBER OF TRANSACTIONS, B: COST PER TRANSACTION IN GAS, C: CUMULATIVE COST IN GAS, D: CUMULATIVE COST IN '$'.

Overall, the cost for the election administrator to run the election with 80 voters and 5 candidates is approximately $\$24.01$. The average cost for each voter to run an election with 80 participants and 5 candidates is approximately $\$12.39$.

The code in the contract 'CryptoContract' is executed locally on the voter's device, not on the Ethereum network. This contract provides the same NIZK proofs for all voters throughout the execution of the protocol.

We have performed experiments with 3, 10, 20, 45, 60, 80 voters and with 2, 3, 4, 5 candidates and plotted the results to show how the costs for the election administrator and voter vary with different numbers of candidates. Figure 2 depicts the average cost for the election administrator based on the number of voters and the number of candidates competing in the election. From this figure, we see that the cost for the election administrator increases linearly with the number of voters. Figure 3 depicts the cost for each voter based on the number of voters participating in the election and the number of candidates competing in the election. From this figure, we see that the cost for each voter remains nearly constant with the number of voters participating in the election.

Figure 4 shows a breakdown of the election administrator's gas cost based on the number of voters when 5 candidates

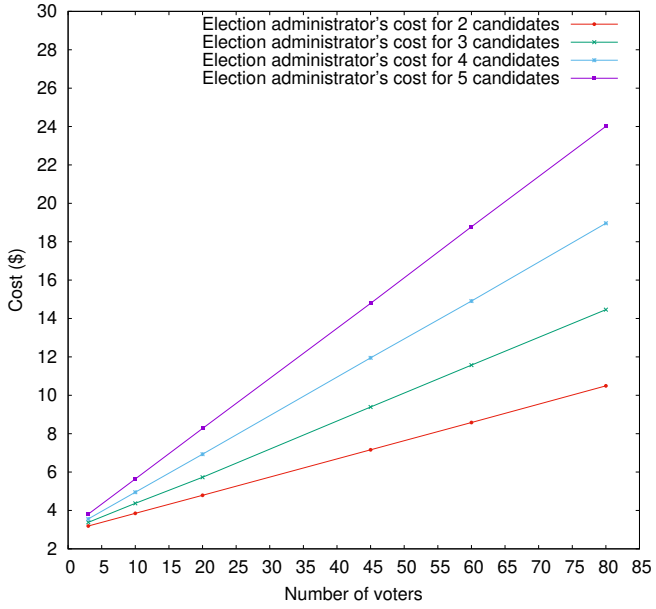

Fig. 2. The election administrator's cost based on the number of voters participating in the election and the number of candidates competing in the election.

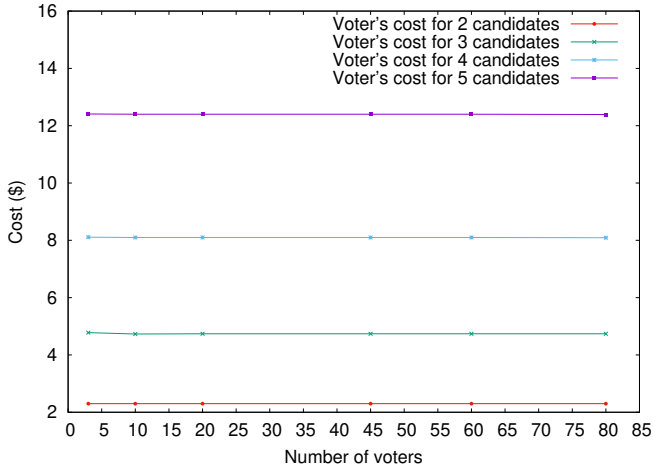

Fig. 3. Each voter's cost based on the number of voters participating in the election and number of candidates competing in the election.

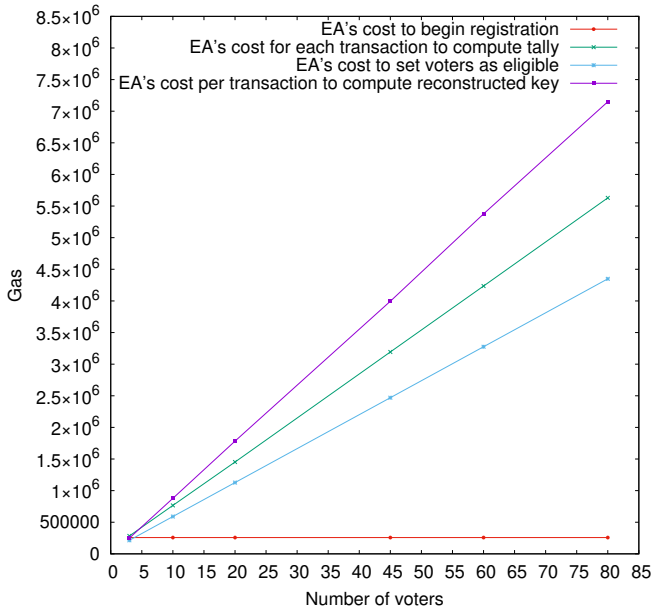

Fig. 4. The gas costs for different tasks of the election administrator (EA) based on the number of voters participating in the election when the number of candidates competing in the election is 5.

are competing in the election. The figure shows that the gas consumption for different tasks increases linearly with the number of voters except for beginning the registration. The maximum gas limit that an Ethereum block can consume was set at 8 million as in June, 2019. In our implementation, the number of transactions to compute reconstructed keys is equal to the number of candidates. Still, each transaction for computing reconstructed keys reaches computation and storage limit for around 87 voters due to the block's gas limit. Due to this block gas limit, we had to make as many transactions as the number of candidates to compute the tally.

Timing measurements analysis. We have performed all the tests on a Chieftec desktop machine running Windows 10 Enterprise version 1809 equipped with 4 cores, 3.10 GHz Intel Core i5-2400 and 32 GB RAM. The timing analysis measurements of different tasks of our protocol are highlighted in Table X. We have rounded all time measurements up to the nearest millisecond. The time measurement of each task is performed using the .call() function on the local daemon.

The voting contract 'VoteContract' contains the main logic of the protocol including the code for enforcing the election process in a timely manner. The task 'Register voting key' in Table X involves the verification of the Schnorr's zero-knowledge proofs created in the first round of the protocol. The time required to complete this action depends on the number of candidates competing in the election. Table X shows the time required to complete this action for different numbers of candidates. Beginning the election involves computing the reconstructed keys for every voter corresponding to each candidate. The time required for this action depends on both the number of voters and the number of candidates competing in the election. Casting a vote involves the verification of the 1-out-of-k zero-knowledge proofs created in the second round of the protocol. It depends on the number of candidates competing in the election. The task 'Tally' in Table X includes calculating the sum of all the cast votes and then getting the tally for each candidate by brute-force (exhaustive searching). The time required for these tasks is shown in Table X for different numbers of candidates contesting in the election.

The cryptography contract 'CryptoContract' is used to create all zero-knowledge proofs using the .call() function. There is no need to send the transactions to this smart contract. The time required to create the schnorr's ZKP in the first round of the protocol depends on the number of candidates competing in the elections, as shown in Table X.

VIII. Discussion on Applications

Ranking based voting systems are popular among voting experts as they tend to gather more information from the voters

Action	A	B	C	D
Create ZKP(X)	34	48	75	91
Register voting keys	57	84	104	138
Begin election	204	305	401	493
Create k 1-out-of-k ZKP	213	515	908	1465
Cast vote	207	483	824	1261
Tally	162	385	457	694

TABLE X
A TIMING MEASUREMENT FOR DIFFERENT FUNCTIONS THAT RUN ON ETHEREUM DAEMON. HERE X IS A k-TUPLE $(x_1, x_2, ..., x_k)$, WHERE k IS THE NUMBER OF CANDIDATES COMPETING IN THE ELECTION. THE k IN THE CREATE-1-OUT-OF-K ZKP REPRESENTS THE NUMBER OF CANDIDATES COMPETING IN THE ELECTION. THE NUMBER OF PARTICIPANTS IS EIGHTY. COLUMNS ARE REPRESENTED AS THE AVERAGE TIME IN MILLISECONDS FOR A: 2 CANDIDATES, B: 3 CANDIDATES, C: 4 CANDIDATES, D: 5 CANDIDATES.

with regard to their preferences as opposed to simple plurality voting where a voter is able to make only a single choice among a set of candidates. It is well known that a plurality voting system can produce election results that do not reflect the true sentiments of the voters [49]. An example of the fact is the 1988 election for electing the prime minister in Canada [49]. The key issue in this election was free trade with the USA. Approximately 60% of the population was opposed to free trade in this three-candidate election. However, the pro free trade candidate won the election since the anti free trade votes got split between two anti free trade candidates. Such pitfalls of the plurality voting system are caused due to the electoral system taking only limited information from voters: only a favorite candidate is indicated. These pitfalls can be avoided by taking more information from voters, e.g., using a ranking-based voting system [49].

Borda count is a typical ranking based voting system that elects a winner taking into account a voter's degree of proclivity toward one or more candidates. Thus, Borda count is sometimes described as a consensus-based voting system [44]. One of the good features of Borda Count is that it is "monotone", as increasing the score for a candidate only helps them win [18]. Hence, this voting system more faithfully reflects the sentiment of the electors than common plurality voting systems. However, these schemes are not heavily deployed in practice due to the complexity involved with them. The complexity around ranking based voting systems have so far led people into shying away from using them in real life. Our paper proposes an easily implementable Borda count e-voting system, and it brings ranking based voting systems closer to practice. Our scheme is publicly verifiable and it guarantees the privacy of every voter to the maximum such that only a full-collusion can break it. Moreover, the scheme does not rely on any trusted third parties. This scheme could motivate researchers to explore new applications in the field of ranking-based e-voting, this previously uncharted territory. Below we provide a list of areas where the Borda Count voting method has been applied.

The Borda count voting has been used to aggregate preferences in many contexts [8]. In addition to democratic elections ([26], [51]), the Borda count voting scheme has been used in elections by several academic institutions and professional bodies [26], [55]. For instance, Borda count is used by X.Org Foundation to elect its board of directors [10]. The Borda count is also used for granting awards in several sports competitions (such as Most Valuable Player Award, Heisman Trophy etc. [50]) and singing competitions (such as Eurovision Song Contest [50]), soccer competitions (such as the RoboCup autonomous robot soccer competition [18]). Borda count is also used by the OpenGL Architecture Review Board as one of the feature-selection methods. It is used as a rank aggregation method for the Web (where voters are the search engines, and candidates are the pages). A summary of these applications of Borda count voting can be found in [65], [21]. In [39], Kijazi and Kant used Borda count method to establish group preferences for alternatives for forest use on Mount Kilimanjaro. Laukkanen et al. [42] and Hiltunen et al. [34] applied several voting methods including Borda count to assess group preferences for forest management plans in Finland. In [11], Burgman et al. discuss potential utility of various voting systems including Borda count for environmental decision making. Borda count is used for Waste Management in several countries [61]. It is used in TOPSIS (technique for order performance by similarity to ideal solution) ranking [56], [58] and an extension of TOPSIS for group decision making [57]. In another context, the Borda count method is widely used for rank aggregation in the information retrieval area [20]. Chatzichristofis et al. [13] propose an image retrieval technique using Borda count.

The public verifiability and freeness from any tallying authorities make our proposed Borda count system suitable for deployment over an Ethereum-like blockchain as we have demonstrated in Section VII. An e-voting system based on blockchain can be effective for corporate governance and shareholder activism [41], [62]. In February 2016, Nasdaq, in cooperation with the Estonian Government, announced a blockchain based e-voting that allows shareholders to vote remotely in Annual General Meetings (AGMs) [9], [41]. Besides Nasdaq, the Abu Dhabi Stock Exchange used blockchain based e-voting to organise shareholder voting in annual general meetings [33], [41]. Compared with these blockchain-based e-voting systems, ours does not require any tallying authorities, so the tallying and the verification of the tallying integrity can be done publicly by the consensus algorithm that underpins the blockchain.

IX. CONCLUSION

In this paper, we have proposed a two-round self-tallying Borda count e-voting scheme. This scheme does not require any trusted party to compute the tally. Instead, the scheme ensures that anyone can compute the tally from the public information made available on the bulletin board. Our scheme ensures the maximum voter privacy, and upon the successful completion of the protocol, the voters are strictly limited to learn only the tally of the election and their own inputs. We have presented security proofs to prove the security of the protocol. Further, the scheme offers public verifiability. Every voter generates NIZK proofs to prove that they have been faithfully following the protocol specification without revealing their secret input. We have implemented the scheme on the

Ethereum blockchain. Both the theoretic and the experimental analysis results show that this scheme is feasible to be used in practice. In future work, we plan to investigate extending this work to support more complex ranked choice voting systems such as STV and Condorcet in a decentralized setting.

Open source code

The source code for the proof-of-concept implementation of the proposed Borda count voting system over the Ethereum blockchain can be found at https://github.com/smartcontract68/Borda_count_smart_contract.

References

[1] Ben Adida. Helios: Web-based Open-audit Voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[2] Ben Adida, Olivier De Marneffe, Olivier Pereira, Jean-Jacques Quisquater, et al. Electing a university president using open-audit voting: Analysis of real-world use of Helios. *EVT/WOTE*, 9(10), 2009.

[3] Ben Adida and Ronald L. Rivest. Scratch & Vote: Self-contained Paper-based Cryptographic Voting. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, pages 29–40, New York, NY, USA, 2006. ACM.

[4] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems. *Real-world electronic voting: Design, analysis and deployment*, pages 171–218, 2016.

[5] Samiran Bag, Muhammad Ajmal Azad, and Feng Hao. E2E Verifiable Borda Count Voting System without Tallying Authorities. In *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019.*, pages 11:1–11:9. ACM, 2019.

[6] Susan Bell, Josh Benaloh, Michael D. Byrne, Dana Debeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, Washington, D.C., August 2013. USENIX Association.

[7] Mihir Bellare and Phillip Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[8] Duncan Black. *The Theory of Committees and Elections*. University Press, Cambridge, 1958.

[9] P. Boucher. What if blockchain technology revolutionised voting? http://www.europarl.europa.eu/RegData/etudes/ATAG/2016/581918/EPRS_ATA(2016)581918_EN.pdf, Sept. 2016. Accessed on 14th Oct., 2019.

[10] Robert Bredereck, Jiehua Chen, Piotr Faliszewski, André Nichterlein, and Rolf Niedermeier. Prices Matter for the Parameterized Complexity of Shift Bribery. *Inf. Comput.*, 251(C):140–164, December 2016.

[11] MARK A. BURGMAN, HELEN M. REGAN, LYNN A. MAGUIRE, MARK COLYVAN, JAMES JUSTUS, TARA G. MARTIN, and KRIS ROTHLEY. Voting Systems for Environmental Decisions. *Conservation Biology*, 28(2):322–332, 2014.

[12] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 410–424, London, UK, UK, 1997. Springer-Verlag.

[13] Savvas A. Chatzichristofis, Konstantinos Zagoris, Yiannis Boutalis, and Avi Arampatzis. A Fuzzy Rank-Based Late Fusion Method for Image Retrieval. In Klaus Schoeffmann, Bernard Merialdo, Alexander G. Hauptmann, Chong-Wah Ngo, Yiannis Andreopoulos, and Christian Breiteneder, editors, *Advances in Multimedia Modeling*, pages 463–472, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[14] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: End-to-End Voter-Verifiable Optical- Scan Voting. *IEEE Security & Privacy*, 6(3):40–46, May 2008.

[15] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

[16] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end Verifiability for Optical Scan Election Systems Using Invisible Ink Confirmation Codes. In *Proceedings of the Conference on Electronic Voting Technology*, EVT'08, pages 14:1–14:13, Berkeley, CA, USA, 2008. USENIX Association.

[17] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[18] Jessica Davies, George Katsirelos, Nina Narodytska, Toby Walsh, and Lirong Xia. Complexity of and algorithms for the manipulation of Borda, Nanson's and Baldwin's voting rules. *Artificial Intelligence*, 217:20 – 42, 2014.

[19] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[20] D. Ding, Le Chen, J. Li, and B. Zhang. AP-Scored Borda Counting for Information Retrieval. In *The Proceedings of the Multiconference on "Computational Engineering in Systems Applications"*, volume 2, pages 1473–1478, Oct 2006.

[21] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. Association for Computing Machinery.

[22] Peter Emerson. *Designing an All-Inclusive Democracy: Consensual Voting Procedures for Use in Parliaments, Councils and Committees*. Jan 2007.

[23] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[24] Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and System Definition of a High-Integrity Election System. In *Workshop on Trustworthy Election. 2006*, 2006.

[25] Followmyvote. Follow my vote. https://followmyvote.com/, 2012. Accessed on 14th Oct., 2019.

[26] Jon Fraenkel and Bernard Grofman. The Borda Count and its real-world alternatives: Comparing scoring rules in Nauru and Slovenia. *Australian Journal of Political Science*, 49(2):186–205, 2014.

[27] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-Round Secure Multiparty Computation. In *Proceedings of the 22Nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 178–193, Berlin, Heidelberg, 2002. Springer-Verlag.

[28] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[29] Jens Groth. Efficient Maximal Privacy in Boardroom Voting and Anonymous Broadcast. In Ari Juels, editor, *Financial Cryptography, 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers*, volume 3110 of *Lecture Notes in Computer Science*, pages 90–104. Springer, 2004.

[30] Feng Hao, Peter Y. A. Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4:62–67, 2010.

[31] Feng Hao and Piotr Zielinski. A 2-Round Anonymous Veto Protocol. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols, 14th International Workshop, Cambridge, UK, March 27-29, 2006, Revised Selected Papers*, volume 5087 of *Lecture Notes in Computer Science*, pages 202–211. Springer, 2006.

[32] A. Hertig. The First Bitcoin Voting Machine Is On Its Way. http://motherboard.vice.com/read/the-first-bitcoin-voting-machine-ison-its-way, Nov. 2015. Accessed on 14th Oct., 2019.

[33] S. Higgins. Abu Dhabi Stock Exchange Launches Blockchain Voting. http://www.coindesk.com/abu-dhabi-exchange-blockchain-voting/, Oct. 2016. Accessed on 14th Oct., 2019.

[34] Veikko Hiltunen, Jyrki Kangas, and Jouni Pykäläinen. Voting methods in strategic forest planning - Experiences from Metsähallitus. *Forest Policy and Economics*, 10(3):117–127, 2008.

[35] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections, New Directions in*

Electronic Voting, volume 6000 of *Lecture Notes in Computer Science*, pages 37–63. Springer, 2010.

[36] Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng Hao. A Fair and Robust Voting System by Broadcast. In Manuel J. Kripp, Melanie Volkamer, and Rüdiger Grimm, editors, *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, volume 205 of *LNI*, pages 285–299. GI, 2012.

[37] A. Kiayias, M. Korman, and D. Walluck. An Internet Voting System Supporting User Privacy. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 165–174, Dec 2006.

[38] Aggelos Kiayias and Moti Yung. Self-tallying Elections and Perfect Ballot Secrecy. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 141–158. Springer, 2002.

[39] Martin Herbert Kijazi and Shashi Kant. Forest stakeholders' value preferences in Mount Kilimanjaro, Tanzania. *Forest Policy and Economics*, 12(5):357–369, 2010.

[40] Ranjit Kumaresan and Iddo Bentov. How to Use Bitcoin to Incentivize Correct Computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 30–41, New York, NY, USA, 2014. ACM.

[41] Anne Lafarre and Christoph Van der Elst. Blockchain Technology for Corporate Governance and Shareholder Activism. *SSRN Electronic Journal*, Jan 2018.

[42] Sanna Laukkanen, Teijo Palander, Jyrki Kangas, and Annika Kangas. Evaluation of the multicriteria approval method for timber-harvesting group decision support. *Silva Fennica*, 39(2):249–264, 2005.

[43] A. K. Lenstra and H. W. Lenstra, Jr. Handbook of Theoretical Computer Science (Vol. A). chapter Algorithms in Number Theory, pages 673–715. MIT Press, Cambridge, MA, USA, 1990.

[44] David Lippman. Voting Theory. *Math in Society*, 2013. Accessed on 17th January, 2020.

[45] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making Smart Contracts Smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, New York, NY, USA, 2016. ACM.

[46] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A Smart Contract for Boardroom Voting with Maximum Voter Privacy. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 357–375. Springer, 2017.

[47] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[48] C. Andrew Neff. Practical High Certainty Intent Verification for Encrypted Votes, 2004.

[49] Jill Van Newenhizen. The borda method is most likely to respect the condorcet principle. *Economic Theory*, 2(1):69–83, 1992.

[50] E. Niou and P.C. Ordeshook. *Strategy and Politics: An Introduction to Game Theory*. EBL-Schweitzer. Taylor & Francis, 2015.

[51] Benjamin Reilly. Social Choice in the South Seas: Electoral Innovation and the Borda Count in the Pacific Island Countries. *International Political Science Review*, 23(4):355–372, 2002.

[52] C. Reitwiessner. Smart contract security. https://blog.ethereum.org/2016/06/10/smart-contract-security/, June 2016.

[53] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt á voter Voter: a Voter-Verifiable Voting System. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, Dec 2009.

[54] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptol.*, 4(3):161–174, January 1991.

[55] Siamak F Shahandashti. Electoral Systems Used around the World. In *Real-World Electronic Voting (Eds. Hao, Ryan)*, pages 93–118. CRC Press, 2016.

[56] Hsu-Shih Shih, Wen-Yuan Lin, and ES Lee. Group decision making for TOPSIS. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, pages 2712–2717. IEEE, 2001.

[57] Hsu-Shih Shih, Huan-Jyh Shyur, and E. Stanley Lee. An extension of TOPSIS for group decision making. *Mathematical and Computer Modelling*, 45(7):801 – 813, 2007.

[58] Hsu-Shih Shih, Chih-Hung Wang, and E.S. Lee. A multiattribute GDSS for aiding problem-solving. *Mathematical and Computer Modelling*, 39(11):1397 – 1412, 2004.

[59] Yonatan Sompolinsky and Aviv Zohar. Secure High-Rate Transaction Processing in Bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 507–527. Springer, 2015.

[60] Pavel Tarasov and Hitesh Tewari. Internet Voting Using Zcash. *IACR Cryptology ePrint Archive*, 2017:585, 2017.

[61] Bojana Tot, Goran Vujić, Zorica Srđević, Dejan Ubavin, and Mário Augusto Tavares Russo. Group assessment of key indicators of sustainable waste management in developing countries. *Waste Management & Research*, 35(9):913–922, 2017.

[62] Christoph Van der Elst and Anne Lafarre. Blockchain and Smart Contracting for the Shareholder Community. *European Business Organization Law Review*, 20(1):111–137, Mar 2019.

[63] B. Wire. Now You Can Vote Online with a Selfie. http://www.businesswire.com/news/home/20161017005354/en/VoteOnline-Selfie, Oct. 2016. Accessed on 14th Oct., 2019.

[64] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger EIP-150 REVISION (759dccd - 2017-08-07), 2017. Accessed on: 2018-01-03.

[65] Y. Zhang, W. Zhang, J. Pei, X. Lin, Q. Lin, and A. Li. Consensus-Based Ranking of Multivalued Objects: A Generalized Borda Count Approach. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):83–96, Jan 2014.

[66] Zhichao Zhao and T.-H. Hubert Chan. How to Vote Privately Using Bitcoin. In Sihan Qing, Eiji Okamoto, Kwangjo Kim, and Dongmei Liu, editors, *Information and Communications Security - 17th International Conference, ICICS 2015, Beijing, China, December 9-11, 2015, Revised Selected Papers*, volume 9543 of *Lecture Notes in Computer Science*, pages 82–96. Springer, 2015.

APPENDIX

APPENDIX A: THE NIZK PROOF ALGORITHMS USED IN THE PROPOSED BORDA COUNT PROTOCOL

Algorithm 1: A prover with identifier ID generates a NIZK proof of knowledge of a secret x such that $(\Gamma' = g^x)$ for known ID, Γ', g.

Input : ID, Γ', g, x such that $(\Gamma' = g^x)$
Output: $\eta = P_K\{x\colon (\Gamma' = g^x)\}$
begin
  choose random $w \in \mathbb{Z}_q$
  calculate
  $t_1 = g^w$.
  calculate
  $c = H(ID, g, \Gamma', t_1)$
  calculate $r = w - cx$
  return $\eta = (r, t_1)$
end

In this section, we present the NIZK proof algorithms that are required in the first and second round of the protocol. Algorithm 1 (resp. Algorithm 2) represents the prover algorithm (resp. verifier algorithm) for generation (resp. verification) of the NIZK proof required in the first round of the protocol. Let us assume that there are k candidates contesting in the election. Algorithm 3 (resp. Algorithm 4) represents the prover algorithm (resp. verifier algorithm) for generation (resp. verification) of 1-out-of-k zero-knowledge proof required in the second round of the protocol. Algorithm 3 (resp. Algorithm 4) is written for the i-th voter V_i to prove (resp. verify) a proposition of the form $\vee_{l=1}^{k}((\Gamma'_l = g^{x_{il}}) \wedge (\Gamma''_l / g^{v_m} = \{g^{y_{il}}\}^{x_{il}}))$, where v_m is the score associated to the m-th rank,

Algorithm 2: Verification of proof η generated by *Algorithm 1* given ID, Γ', g.

Input : $ID, \Gamma', g, \eta = (r, t_1)$
Output: success or failure
begin
  calculate
  $c = H(ID, g, \Gamma', t_1)$
  calculate
  $t'_1 = g^r \Gamma'^c$
  if $t_1 = t'_1$ **then**
    return success
  else
    return failure
  end
end

Algorithm 3: A prover with identifier ID generates a proof of knowledge of a secret x_{ij} such that $\vee_{l=1}^{k}((\Gamma'_l = g^{x_{il}}) \wedge (\Gamma''_l/g^{v_m} = \{g^{y_{il}}\}^{x_{il}}))$, where v_m is the score associated to the m-th rank, $m \in \{1, 2, ..., k\}$, and the score v_m is given to the j-th candidate, $j \in \{1, 2, ..., k\}$.

Input : $ID, g, k, (\Gamma'_l, \Gamma''_l, \{g^{y_{il}}\})_{l=1}^{k}, x_{ij}, j, g^{v_m}$ such that $\Gamma'_j = g^{x_{ij}}$ and $\Gamma''_j/g^{v_m} = \{g^{y_{il}}\}^{x_{ij}}$
Output: $\Pi_m = P_K\{x_{ij} : \vee_{l=1}^{k}((\Gamma'_l = g^{x_{il}}) \wedge (\Gamma''_l/g^{v_m} = \{g^{y_{il}}\}^{x_{il}}))\}$
begin
  choose random
  $w, r_1, c_1, r_2, c_2, ..., r_{j-1}, c_{j-1}, r_{j+1}, c_{j+1}, ..., r_k, c_k \in \mathbb{Z}_q$
  calculate $t_{11} = g^{r_1}\{\Gamma'_1\}^{c_1}, t_{12} = \{g^{y_{i1}}\}^{r_1}\{\Gamma''_1/g^{v_m}\}^{c_1}, t_{21} = g^{r_2}\{\Gamma'_2\}^{c_2}, t_{22} = \{g^{y_{i2}}\}^{r_2}\{\Gamma''_2/g^{v_m}\}^{c_2}, ..., t_{j-11} = g^{r_{j-1}}\{\Gamma'_{j-1}\}^{c_{j-1}}, t_{j-12} = \{g^{y_{ij-1}}\}^{r_{j-1}}\{\Gamma''_{j-1}/g^{v_m}\}^{c_{j-1}}, t_{j1} = g^w, t_{j2} = \{g^{y_{ij}}\}^{w}, t_{j+11} = g^{r_{j+1}}\{\Gamma'_{j+1}\}^{c_{j+1}}, t_{j+12} = \{g^{y_{ij+1}}\}^{r_{j+1}}\{\Gamma''_{j+1}/g^{v_m}\}^{c_{j+1}}, ..., t_{k1} = g^{r_k}\{\Gamma'_k\}^{c_k}, t_{k2} = \{g^{y_{ik}}\}^{r_k}\{\Gamma''_k/g^{v_m}\}^{c_k}$
  calculate
  $c = H(ID, (g, \Gamma'_l, \{g^{y_{il}}\}, \{\Gamma''_l/g^{v_m}\})_{l=1}^{k}, (t_{l1}, t_{l2})_{l=1}^{k})$,
  calculate
  $c_j = c - (c_1 + c_2 + ... + c_{j-1} + c_{j+1} + ... + c_k)$
  calculate $r_j = w - c_j x_{ij}$
  return $\Pi_m = (c_1, c_2, ..., c_{j-1}, c_j, c_{j+1}, ..., c_k, r_1, r_2, ..., r_{j-1}, r_j, r_{j+1}, ..., r_k, (t_{l1}, t_{l2})_{l=1}^{k})$
end

Algorithm 4: Verification of proof Π_m generated by *Algorithm 3* given $ID, g, k, (\Gamma'_l, \Gamma''_l, \{g^{y_{il}}\})_{l=1}^{k}, g^{v_m}$, where v_m is the score associated to the m-th rank. However, the verifier does not know to which candidate (i.e. j) the score v_m is given.

Input : $ID, g, k, (\Gamma'_l, \Gamma''_l, \{g^{y_{il}}\})_{l=1}^{k}, g^{v_m}, \Pi_m = (c_1, c_2, ..., c_k, r_1, r_2, ..., r_k, (t_{l1}, t_{l2})_{l=1}^{k})$
Output: success or failure
begin
  calculate
  $c' = H(ID, (g, \Gamma'_l, \{g^{y_{il}}\}, \{\Gamma''_l/g^{v_m}\})_{l=1}^{k}, (t_{l1}, t_{l2})_{l=1}^{k})$
  if $(c' \neq (c_1 + c_2 + ... + c_k))$ **then**
    return failure
  end

  calculate
  $t'_{11} = g^{r_1}\{\Gamma'_1\}^{c_1}, t'_{12} = \{g^{y_{i1}}\}^{r_1}\{\Gamma''_1/g^{v_m}\}^{c_1}, t'_{21} = g^{r_2}\{\Gamma'_2\}^{c_2}, t'_{22} = \{g^{y_{i2}}\}^{r_2}\{\Gamma''_2/g^{v_m}\}^{c_2}, ...,$
  $t'_{k1} = g^{r_k}\{\Gamma'_k\}^{c_k}, t'_{k2} = \{g^{y_{ik}}\}^{r_k}\{\Gamma''_k/g^{v_m}\}^{c_k}$
  if $((t_{11} = t'_{11})\&\&(t_{12} = t'_{12})\&\&(t_{21} = t'_{21})\&\&(t_{22} = t'_{22})\&\&...\&\&(t_{k1} = t'_{k1})\&\&(t_{k2} = t'_{k2}))$ **then**
    return success
  else
    return failure
  end
end

$m \in \{1, 2, ..., k\}$, the score v_m is given to the j-th candidate, $j \in \{1, 2, ..., k\}$, Γ'_l represents X_{il} corresponding to the l-th candidate in the first round of the protocol, and Γ''_l represents Z_{il} corresponding to the l-th candidate in the second round of the protocol as discussed in section IV-A. The algorithm for the case when the score v_m is given to any candidate other than the j-th candidate can be obtained by straightforward modifications. The symbol 'ID' denotes the publicly known identifier of the voter. Following [31], [30], we include the voter's ID in the hash for the Fiat-Shamir transformation to bind the identity with the ZKP and to prevent replay attacks. For example, in our implementation over Ethereum, we use the sender's unique identity (msg.sender) as 'ID' in the argument of the hash function. The purpose of using the sender's unique identity (msg.sender) is already discussed in section VII-D. The symbols Γ' and x in the Algorithm 1 and Algorithm 2 represent X_{ij} and x_{ij} respectively for each of the i-th voter V_i in the first round of the protocol as described in section IV-A, where $j \in \{1, 2, ..., k\}$.