applied sciences

MDPI

Article

Privacy-Preserving E-Voting System Supporting Score Voting Using Blockchain

Ali Alshehri [1,*], Mohamed Baza [2], Gautam Srivastava [3,4,5], Wahid Rajeh [6], Majed Alrowaily [1] and Majed Almusali [6]

1 Department of Computer Science, University of Tabuk, Tabuk 71491, Saudi Arabia
2 Department of Computer Science, College of Charleston, Charleston, SC 29424, USA
3 Department of Math and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada
4 Research Centre for Interneural Computing, China Medical University, Taichung 40402, Taiwan
5 Department of Computer Science and Math, Lebanese American University, Beirut 1102, Lebanon
6 Department of Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia
* Correspondence: a.alshehri@ut.edu.sa

Abstract: With the advancement of cyber threats, blockchain technology has evolved to have a significant role in providing secure and reliable decentralized applications. One of these applications is a remote voting system that allow voters to participate in elections remotely. This work proposes a privacy-preserving e-voting system supporting score voting using blockchain technology. The main challenge with score voting compared to the regular yes/no voting approach is that a voter is allowed to assign a score from a defined range for each candidate. To preserve privacy, votes shall be encrypted before submission to the Blockchain, however, a malicious voter can modify the score value before encrypting it to manipulate the elections result for the favor of a certain candidate. To address this challenge, the proposed scheme allows voters to first prove that the submitted score lies in the predefined range before the vote is added to the Blockchain to ensure fairness of the election. The performance of our scheme is evaluated against a set of comprehensive experiments designed to determine optimal bounds for workload and transaction send rates and measure the impact of exceeding these bounds on critical performance metrics. The results of these simulations and their implications therefore indicate that the proposed scheme is secure while being able to handle up to 10,000 transactions at a time.

Keywords: security; privacy; e-voting; blockchain; score-voting

Citation: Alshehri, A.; Baza, M.; Srivastava, G.; Rajeh, W.; Alrowaily, M.; Almusali, M. Privacy-Preserving E-Voting System Supporting Score Voting Using Blockchain. *Appl. Sci.* **2023**, *13*, 1096. https://doi.org/10.3390/app13021096

Academic Editors: Petr Dzurenda, Sara Ricci and Jordi Castellà-Roca

Received: 29 November 2022
Revised: 28 December 2022
Accepted: 4 January 2023
Published: 13 January 2023

1. Introduction

With the advancement of cyber threats, blockchain technology has evolved to have a significant role in providing secure and reliable decentralized applications. The most indispensable operations within any democratic government, and the means by which these democracies flourish, are its elections. Electronic voting (e-voting) systems allow individuals to conduct votes and elections remotely [1]. In the wake of a global pandemic, the need for a remote voting alternative became especially pronounced, as the existing alternative, mail-in paper ballots, raised concerns over their ability to be easily manipulated. A digital voting platform can reduce or eliminate the need for these face-to-face systems, paper ballots, and/or the use of a third party (such as an email service provider or the USPS). The use of e-voting systems is not solely limited to political usage and can be used for corporation leadership and decision-making, and within distributed Internet of Things (IoT) networks for both decision-making and leadership [2,3]. Some countries such as Estonia have already implemented an online voting system for its national binding elections that is available to its entire electorate and remains the first country and one of the only countries to do so [4,5].

Recently, cryptographic primitives such as ElGamal encryption schemes, blind signatures, ring signatures, and homomorphic encryption have been implemented within contemporary online voting systems to guarantee voter privacy and anonymity, ballot secrecy, and the verifiability of election results. Despite these measures, however, security incidents have still occurred in online election systems. In 2015, the New South Wales (NSW) Electoral Commission admitted that up to 66,000 election votes had been compromised on the online iVote website [6], where the compromised votes could have been exposed or tampered with unnoticed. The Pentagon refuses to endorse electronic voting; and in May 2020, the FBI and NIST, among other federal agencies, released a warning to the states that the return of marked ballots online was a "high risk" venture [7]. Voatz, perhaps currently the most widespread e-voting system in use in the United States, was found to have "concerning" security issues, leading to the notification of the DHS of these security issues [8].

Blockchain technology and the concept of decentralized applications (DApps) lend special potential to the field of remote e-voting. Namely, its ability to provide accountability through verification and security and robustness through immutability and their distributed nature enable an election to take place with minimal interaction of a third party during the process, thus preserving election integrity and removing the potential for bias. In simpler terms, a blockchain is merely a novel data structure consisting of interconnected blocks called nodes. Each node contains an archive of a transaction that must be committed by the network of peer devices, called miners, via a consensus algorithm. The addition of smart contracts to blockchain enabled the shift of blockchain utilization from solely for cryptocurrency to DApps as well. Smart contracts are simply programs that contain the business logic for the application and are stored on-chain [9–11]; it is these contracts that enable the application to execute and enforce without the need for a central third party.

Despite the number of existing schemes that have been proposed, many still suffer from several shortcomings such as lack of fairness and high computational and communication overheads to preserve privacy. This paper introduces a protocol for an e-voting supporting score-voting system using blockchain technology. The paper addresses the challenge between the open nature of Blockchain and privacy. To preserve privacy, voters will send their votes encrypted to the Blockchain. However, a malicious voter may modify the score value for a specific candidate to manipulate the elections result in favor of a certain candidate. This is a serious problem, especially in the case of score voting compared to the regular yes/no voting approach, because in score voting a voter is allowed to assign a score from a defined range for each candidate. The proposed approach ensures the security against that malicious behavior by leveraging the zero knowledge set membership proof so that the voter shall prove first in a zero knowledge that the selected score lies in the predefined set without allowing the Blockchain and/or eavesdroppers to know the score value per candidate. In addition, the score value is masked to preserve privacy. Finally, to obtain the election results, all participants including the election authority can tally the election results in a decentralized manner. We also conduct an empirical evaluation of our protocol on the Ethereum Blockchain, and find that it is able to sustain workloads up to 10,000 transactions sent at a rate of 200 per second before the system begins to experience significant declines in performance metrics. Such results suggest that, if deployed on the Ethereum network, our scheme is ideal for small and mid-sized elections and other selection processes of under 10,000 participants.

The remainder of this paper is organized as follows. In Section 2, we discuss the most recent related works. Section 3 outlines the preliminaries used in this research work. In Section 4, the system/threat models, as well as the components of the proposed scheme are presented. Section 5 lays out the detailed construction of the proposed scheme. Then, Section 6 discusses the experiments and results to evaluate the scalability and efficiency of the proposed scheme. Finally, Section 8 concludes the paper and the findings of this research.

2. Related Work

In this section, we review existing works aimed at building a secure e-voting system. A summary of the most recent e-voting protocols is given in Table 1.

Table 1. A summary of the most recent e-voting protocols.

Article	Year	Problem(s) Solved	Main Approach	Advantages (+) / Limitations (−)
Electanon [12]	2022	Allows for multiple types of ranked choice e-voting.	Implements libraries for several RC voting algorithms, Merkle trees for voter registration, and Semaphore for zkSNARK implementation.	(+) Flexible system allows for multiple ranked-choice protocols. (−) Ranked choice (RC) algorithms are inefficient.
Score-based self-tallying E-voting system [13]	2021	Incorporates the score-based voting model without compromising the security of other Blockchain models.	Ledger-based model that uses ElGamal encryption and a novel dual-ZKP method to validate range-based score voting.	+) Voter privacy is not compromised. (−) There is no addressing of coercive issues.
Self-Tallying E-voting system [14]	2021	Does not require 100% voter participation in order for a tally to be conducted.	Blockchain system that uses ElGamal and non-interactive ZKP to ensure verifiability. Uses an algorithm that cancels out all ballot-related random numbers so that not each one is required to vote in order to tally the results.	(+) Does not require 100% voter participation. (+) Results can be tallied independently of voter participation. (−) There is no addressing of adaptive issues.
Blockchain-based or cloud-server based e-voting [15]	2021	Maintains usual privacy and security standards on the Blockchain.	Uses a Blockchain-based approach or a centralized model. Utilizes ZKP and secure MPC to achieve privacy during tally.	(+) Decentralized scheme offers high security measures. (−) Centralized model vulnerable to DoS and MITM attacks and single-point-of failure attacks.
A Blockchain-based traceable self-tallying e-voting protocol in the AI era [16]	2020	Solves self-tallying and voter privacy within an AI context	Blockchain network utilizing homomorphic time-lock puzzles and event-oriented linkable group signatures.	(+) Successfully detaches decryption time from being proportional to the number of voters (+) More efficient for large electorates. (−) System is expensive and inefficient for small electorates.
Self-tallying protocol for decentralized IoT [3]	2020	Votes are obscured but verifiable, Results hidden until all votes are cast	Blockchain network utilizing Distributed ElGamal Encryption and Sigma-Protocol ZKP to ensure fairness and maximal ballot secrecy.	(+) All voters have to submit decryption keys in order for the composite key to be computed, preventing early release of results. (−) Requiring all voters to submit partial keys is computationally expensive as the number of voters increase. (−) The system only allows for two candidates.

E-voting in Industry. As early as 2008 several centralized online platforms were proposed; Refs. [17,18] both proposed similar protocols for remote i-voting and direct recording

electronic (DRE) systems, respectively. These systems were already attempting to provide universal verifiability and coercion resistance, and both Refs. [17,18] used non-interactive zero-knowledge proofs (NIZK) and homomorphic encryption schemes to accomplish these goals; but because the architecture is client-server based, they are prone to a number of attacks including denial-of-service (DoS) and man-in-the-middle (MITM). Voatz is the first mobile e-voting application to be used in the United States [19]; however, a group of MIT researchers discovered in Ref. [8] that Voatz, despite their claims of high security and privacy, had some fundamental flaws that enabled several methods of attacks. These flaws are largely in part due to the use of a server to communicate between the voter and the Blockchain. On the client side, they discovered that an attacker who gained physical access to a voter's device could, in the area of the "Internet of Things", with root privileges, disable Voatz's host-based protections. This was as simple as using four lines of code to prevent the third-party anti-malware service from running [8]; once the anti-malware is disabled, the attacker has gained full control over the user's account and is able to steal authentication data as well as control the user's vote. The voter can also complete a Stealth UI Modification Attack that keeps the UI looking as the user expects it too but then alters a vote or carries out other malicious behavior. The researchers also found attacks that could be carried out server-side; the server is capable of altering votes, violating privacy and exposing voter information, and controlling the election outcome via MITM attacks [8].

Self-tallying E-voting Systems. In Ref. [3], the authors have proposed a scheme that utilizes Distributed ElGamal Encryption and Sigma-protocol based Zero Knowledge Proofs to ensure maximal ballot secrecy and fairness. Fairness within the system ensures that a malicious user cannot collect other user's votes and obtain partial results before all voters have participated. The encryption scheme requires all voters to submit their partial decryption keys before the result can be decrypted, thus eliminating the ability for results to be determined without collusion amongst all the voters who are yet to cast a ballot. However, to obtain the vote of a specific voter, all the voters should send their partial decryption keys. This, therefore, makes the system inefficient because such a process requires too much computation and communication overhead, especially when the number of voters increases significantly. The work in Ref. [20] proposed a scheme for a self-tallying system for IoT software updates, similar to the scheme proposed in Ref. [3], which uses commitments, ZKP, blind signatures, and mix-nets to accomplish advanced security and privacy features. The commitments are an important feature that deals with abortive issues that allow for a vote to be "recovered" even if the vote is lost or fails to be cast [20]. However, this system and all self-tallying systems require everyone to vote in order to tabulate results, because when they vote their private decryption keys are used to compute the aggregate decryption key. As such, the system requires a consensus of all remaining voters so that the vote can be released, and it could be an avenue of attack on the system if a malicious coalition were formed. In Ref. [14], another scheme similar to Refs. [3,20] is proposed; however, this scheme seems to be the first to not require all voters to vote in order to tabulate the result (voters can abstain from voting and not obstruct the release of results) by generating random numbers amongst those who are abstaining, which will cancel each other during the tabulation of the ballots. This lends more practicality to the system; however, the adaptive issues fail to be addressed. The proposed approach in Ref. [16] gives a scheme different from the ones proposed above. This scheme proposes different solutions to the same security issues; they leveraged homomorphic time-lock puzzles to implement the self-tally and a new event-oriented linkable group signature to protect voter's privacy and prevent double voting. This enables the decryption time to remain independent of the number of voters; however, the cost is still expensive in terms of communication and computational overhead.

Score Voting. In Ref. [13], a voting system that utilizes score-based voting in a privacy-preserving manner is proposed. The concept of a dual zero-knowledge proof (dual-ZKP) is introduced, as well as the use of ElGamal Encryption. The dual-ZKP guarantees that two constraint conditions on one set of elements can be proven to the verifier. This enables

the voter to prove that (a) their score for each candidate falls within a predefined range, and (b) that the sum of the scores given by the voter is equal to a predefined constant. Additionally, ElGamal encryption was used a lot, as in Ref. [3], to ensure maximal-ballot secrecy, and commitments to deal with abortive issues; in this way, the selection of a voter can be retrieved from their commitment even if they decide to abort their actual vote. This prevents them from "locking" the ballot results and ensures that the results can always be tabulated. However, the scheme fails to deal with coercive issues, meaning it is unable to detect and handle the situation of an impostor using someone's account to alter their vote by casting it in their place. Another scheme was proposed in Ref. [21], which provides a secure e-voting protocol where an independent tallier provides the tabulation. This protocol takes advantage of D-out-of-D sharing and multiparty computations to provide security, while manipulating vectors to provide five types of score-based voting. However, the protocol suffers from a high computation and communication overhead due to some of the score-based algorithms as well as the MPC-based tally scheme.

Blockchain-based E-voting Systems. The proposed approach in Ref. [12] is a scheme that allows for ranked choice voting over a Blockchain network; the scheme achieves security and privacy goals in a similar manner as contemporary research, primarily through a tool Semaphore, which helps implement zkSNARKS, a type of ZKP. The scheme also utilizes Merkle trees for voter registration, and uses libraries to be able to support several RCV methods, including Borda, Majority Rule, and Condorcet. Ref. [22] also proposed a scheme for Borda RCV over Blockchain, but failed to allow the tally to be computed should users abstain from voting. Ref. [12] requires all candidates to be ranked on each ballot, which can be burdensome for the user. In Ref. [15], another scheme is proposed, which can alternatively use a cloud server instead of Blockchain to store the data. Both methods use ZKP and secure MCP to protect voter privacy and enhance system security. Finally, in Ref. [23], a system using Elliptic curve cryptography (ECC) for key and authentication data encryption is used; however, the protocol fails to encrypt the vote itself, resulting in a severe loss of privacy for the voters and the potential for severe election consequences.

3. Background

In this section, we discuss the background of Blockchain technology and some of the cryptosystems that are used in this paper. The primary notations used in this paper are given in Table 2.

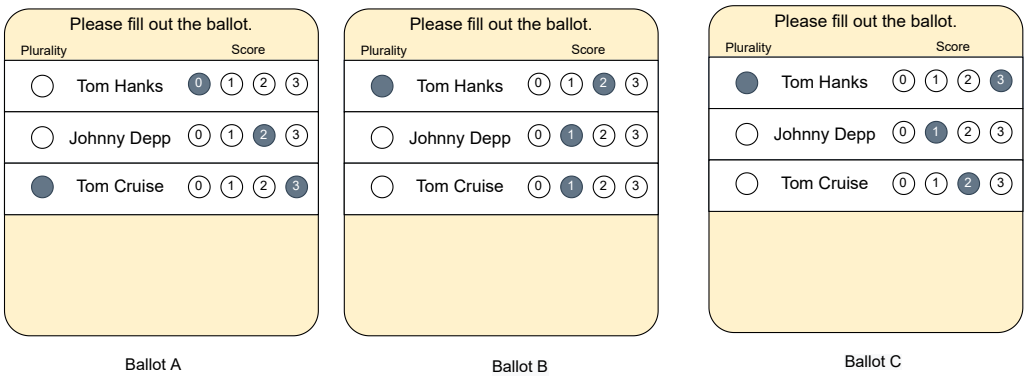
Table 2. Key notations.

Notation	Definition
$\mathcal{EA}$	The Election Administrator
$\mathcal{V}_i, \mathcal{C}_j$	Denotes a Voter or a Candidate
$V_{i,j}$	Denotes a voter's ballot
$C_{i,j}$	Denotes a commitment to a ballot
g	The generator of the elliptic curve group G of prime order q
x_i, y_i	The private/public key pair for voter i where $y_i = g^{x_i}$
n_v, n_c	The number of voters and candidates in the election
$[n_v], [n_c]$	The set of all voters and all candidates
$p_{i,j}$	The score assigned to candidate j by voter i
P_j	The aggregate score for candidate j
Y_i	The composite public key
$x_{i,j}$	The private voting key for each voter i where $j \in [n_c]$

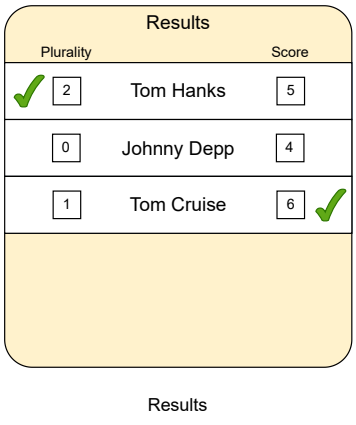
3.1. Score Voting

Score-based score voting is a special type of election procedure that allows voters to assign a score to each candidate. This method has started to become more widely used in various elections, such as the election for the Secretary General of the United Nations and the election of Green Party of Utah [24]. In this method, each voter has the ability to

score each individual candidate within a specified range, such as 0 to 5. After all candidates on a ballot have been scored, the sum of the scores should be exactly equal to a number predefined by the election authority. The result is tabulated by obtaining a summation of the scores for each candidate, with the candidate receiving the highest score being declared the winner. Score voting allows the voters to express preferences of varying degrees for different candidates and has been shown to be more advantageous than plurality voting and ranked choice voting (RCV). According to Ref. [25], *score voting eliminates split votes (where no winner receives a majority) and reduces the number of spoiled ballots (ballots that have been marked or tabulated incorrectly) as compared to plurality and ranked systems.* Moreover, score voting enables a more accurate measure of support amongst losing candidates and is easy to implement on current election systems because it does not require the complex tallying algorithms that normal ranked-choice voting needs. Figure 1 gives an example of a score-based ballot in comparison with a plurality system. Each candidate receives a score from each of the three voters; the candidate who receives the highest score on a single vote is assumed to have won the point in a plurality system. It can be seen that here Tom Hanks received the highest score twice, making him the winner in a traditional plurality system; however, Tom Cruise aggregated the highest score, giving him the victory in a score-based election.



(**a**)



(**b**)

Figure 1. Illustration of a score-based election with three voters and three candidates (**a**). In (**b**), the corresponding plurality vote is visible on the left of each ballot. Notice how plurality voting yields a deceptive result when compared to the score-based result. (**a**) Example of three different votes in a score-based election. The ballot includes three candidates namely Tom Hanks, Johnny Depp, and Tom Cruise. The ballot includes a score of four per candidate. The voter has to select one score per each candidate. (**b**) The results of the score-based and regular elections.

3.2. Blockchain and Smart Contracts

A Blockchain is a data structure that maintains unique properties that make it a public and transparent ledger. This ledger is comprised of a sequence of blocks that archive transactions that occur on the Blockchain [26]. Each new transaction is approved by nodes within the chain, called miners, by means of a consensus algorithm. Only new transactions can be added; existing ones cannot be altered or removed, giving the network an immutable, append-only quality. Another key characteristic of Blockchain is its transparency because every detail about a transaction is stored on the Blockchain in such a manner that it is visible to all network peers [27]. Originally, Blockchain was constrained to be used for cryptocurrencies, but smart contracts have enabled Blockchain's use to be expanded to almost any application. Such a Blockchain-enabled application is called a decentralized application (DApp). The smart contract can be considered a program that contains the business rules and logic for the application [28]. Each node within the chain runs the smart contract in order to maintain and update their local copies of the Blockchain, according to the execution results of the smart contract [29]. The smart contract, which is stored within the Blockchain, eliminates the need for a third party to execute or manage the network, lending the Blockchain its decentralized quality [30].

3.3. Zero-Knowledge Proof of Set Membership

A set membership proof enables a prover to prove that a secret value lies within a given public set. This proof is done in a zero-knowledge manner such that no information about the value is made public. This proves particularly useful in e-voting, particularly score-based since a voter is required to prove *(a)* that their vote belongs to the set of approved candidates; *(b)* that the score they give to a candidate falls within the valid range (a set); and *(c)* that they are indeed a member of the set of registered voters eligible to vote in the active election. Using the Camenisch and Stadler [31] notation for proofs of knowledge (PoK):

$$PK(\delta, \gamma) : Y = g^{\delta} h^{\gamma} \wedge (\gamma \in \phi)$$

where $Y = g^{\delta} h^{\gamma}$ is a Pedersen commitment of the integer *more symbols* using randomness γ. This proof convinces the verifier that the secret committed in Y lies in the set ϕ without explicitly revealing ϕ in the proof. The set ϕ can be an input that is common both to the verifier and the prover. The proof can also be made non-interactive by implementing with the Fiat-Shamir heuristic.

This type of proof has three security guarantees: *(i) Soundness.* An honest verifier cannot be convinced by any prover if that prover did not compute the results correctly. *(ii) Completeness.* The honest prover will always convince an honest verifier that a true statement is indeed valid. *(iii) Zero-knowledge.* The verifier learns nothing during the proof distribution other than the fact a statement is true (or false).

4. System and Security Models

This section discusses the network and security models of the proposed scheme.

4.1. Network Model

The following entities are considered in the network as illustrated in Figure 2.

- *Blockchain.* The Blockchain network is the cornerstone element of our system. The Blockchain is responsible for handling all voting transactions. All business logic is stored and executed on-chain in the system smart contracts;
- *Election Authority.* The election authority can be any entity conducting an election or holding a decision-making vote; this entity is not required to have political goals. An election authority could be a small non-profit seeking to make an important decision amongst its board-members, or it could be a candidate participating in an federal election, or it could be anywhere in between. The election authority is responsible

for publishing an election contract to the Blockchain. They are also responsible for maintaining the list of candidates and the list of eligible voters;

- *Voter.* The voter is an entity that is making and submitting a selection in the election. The voter must be authorized (i.e., registered) in order to cast a vote. Users who cannot present sufficient proof of registration or authorization must be denied access to the system. The voter is able to verify their ballot and the overall results of the election.

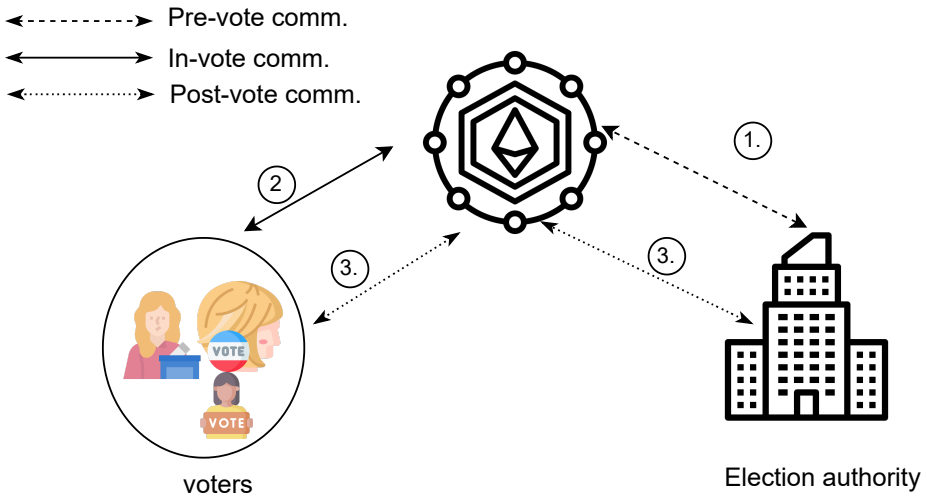


Figure 2. System architecture: (1) An organization publishes an election contract to the Blockchain. (2) The voter casts his/her vote to the Blockchain. (3) The Blockchain computes the results of the election and publishes the results to the candidates and voters.

4.2. Threat Model

We follow the standard Blockchain threat model presented in Ref. [32], where the Blockchain is trusted for execution correctness and availability, but not for privacy. Two types of adverse attackers are considered: passive attackers and active attackers. The passive adversaries are not actively engaged in the process of voting, but instead are trying to gain knowledge of the election proceedings by eavesdropping on communication channels and the Blockchain. Active adversaries, on the other hand, are actively involved in the proceedings, attempting to manipulate the election process. The smart-contract code is visible and checkable by anyone once it is deployed and is guaranteed to work as specified, free from tampering. Likewise, any data submitted and stored to the contract can be directly read by all parties of the system as well as any external curious users. In addition, the following threats are also considered:

- Global eavesdroppers can read all transactions recorded on the Blockchain to learn their voting records;
- A voter may cheat by using a score value that is not included in the ballot to increase the chances for one candidate to win the elections.

A self-tallying score voting system must satisfy the following security requirements. All four requirements are assumed to be fulfilled for partial collusion only, meaning that in a case of full collusion against one voter, that voter's privacy will be breached.

- *Maximum ballot secrecy .* Any colluding attackers can only learn the partial tally with the consent of remaining voters;
- *Self-tallying.* The final tally can be computed once the voting phase is finished. Any voter or third-party auditor/observer may compute the election results;
- *Fairness.* Strongly related to the first requirement, fairness states that no single user should have a priority to obtain a partial tally before the results are ready to be computed. Fairness issues stem from active adversaries who carry out attacks by exploiting abortive or adaptive issues. An abortive voter refuses to reveal their vote

and hence obstructs the tally; an adaptive exploiter uses the fact they are the last voter to obtain a final tally before casting their ballot, leading to interference or abortive measures.

5. The Proposed Voting Scheme

In this section, a decentralized and self-tallying score voting system is proposed. At the start of an election cycle, the election administrator $\mathcal{EA}$ publishes a candidate list $\{\mathcal{C}_1, \cdots, \mathcal{C}_{n_c}\}$ and defines a fixed number $\mathcal{P}$ as the maximum evaluation score each candidate can receive. It is a necessary condition that the score assigned to each candidate should be no less than zero and no greater than $\mathcal{P}$; likewise the sum of each score on the ballot must be equal to $\mathcal{P}$. Our scheme, inspired by the Open Vote Network protocol [33], consists of three phases as follows. The *Setup* phase generates the parameters for the system where the $\mathcal{EA}$ is the entity that provides the candidate list as well as a list of eligible voters that they must authenticate. Each voter $\mathcal{V}_i$ generates their secret and public key pair $x_i, y = g^{x_i}$. Then, during the *Vote* phase, each voter $\mathcal{V}_i$ sends his/her encrypted vote to the Blockchain as well as a proof that the submitted score falls within a publicly defined range $[0, P]$. To do this, we use a ZKP to prove that this condition is met. In order to accomplish this, we utilize a zero-knowledge set membership proof for each score to show that the score falls within the range without disclosing the assigned score. Finally, in the *Tally* phase, all the ballots $V_{i,i\in[n_v]}$ are tallied without the need for any entity's secret key, effectively enabling any entity to compute the results.

5.1. Setup Phase

The $\mathcal{EA}$ produces the list of candidates and eligible votes as well as the system parameters. Each voter $\mathcal{V}_i$ produces his/her private/public key pair $x_i, y_i = g^{x_i}$ for $i \in [n_v]$ during the **Register** algorithm. The $\mathcal{EA}$ randomly selects a value $\mathcal{S}$ and divides it into n values and secretly sends each eligible voter, v_i, $\mathcal{S}_{v_i}$. Then, the $\mathcal{EA}$ selects an elliptic curve group (G, g), where g is the generator of the elliptic curve group G of prime order q, and $\mathbb{Z}_q$ is the integer set $\{0, 1, \cdots, q-1\}$ and $\mathbb{Z}^*{}_q = \{1, \cdots, q_1\}$. The $\mathcal{EA}$ also selects a value P that represents the sum of the evaluation scores given to each candidate on the ballot; thus the $\mathcal{EA}$ is also defining a set ϕ where $\phi = [0, 1, 2, \cdots, P]$. The public parameters PP are hence $(G, g, \mathcal{P})$. Finally, the election authority needs to initialize the ZKSM as follows.

1. Defines a set ϕ of k scores where $\phi = \{p_1, \cdots, p_k\}$;
2. Picks a random number $x \in_R \mathbb{Z}_p$ and computes a corresponding public $y \in g^x$, where g is the generator of the order $-q$ subgroup of $\mathbb{Z}_p$;
3. Computes for every element $i \in \phi$ the corresponding signature using g and x: where, $\mathcal{A}_i = g^{\frac{1}{(x+i)}}$;
4. Publishes a new smart-contract with the ϕ.

Finally, the $\mathcal{EA}$ also publishes a list of candidates $(\mathcal{C}_1, \cdots, \mathcal{C}_{[n_c]})$ and list of eligible voters $(\mathcal{V}_1, \cdots, \mathcal{V}_{[n_v]})$ to the Blockchain.

Each voter $\mathcal{V}_i$ randomly selects their private key $x_i \in \mathbb{Z}^*{}_q$ and computes their public key $y_i = g^{x_i}$, with $i \in [n_v]$. Subsequently, $\mathcal{V}_i$ generates a non-interactive zero-knowledge proof to show that they properly computed their keys: $ZKPoK_1\{(x_i) : y_i = g^{x_i}\}$. The prover $(\mathcal{V}_i)$ knows the secret key x and public key $y = g^x$, but needs to prove that they know x. The prover selects another random $w_i \in_R \mathbb{Z}_q$ and calculates $a_i = g^{w_i}$. Using the Fiat–Shamir heuristic, a challenge c is issued via a secure cryptographic hash function so that $c_i = H(g, y_i, a_i) = H(g, g^{x_i}, g^{w_i})$. The user then computes $r_i = w_i - x_i c_i$ and sends the proof to the Blockchain. If a_i can be verified to be equal to $g^{r_i} {y_i}^{c_i}$, then the proof is accepted. In addition to their decryption keys, each voter also needs a n_c keys $(x_{i,1}, x_{i,2}, \cdots, x_{i,n_c})$. Each voter $\mathcal{V}_i$ submits the corresponding public keys $(y_{i,1}, y_{i,2}, \cdots, y_{i,n_c}) = (g^{x_{i,1}}, g^{x_{i,2}}, \cdots, g^{x_{i,j}})$ and a corresponding ZKP $\pi_{i,j}$ for each key.

5.2. Voting Phase

During the *Vote* phase, each voter $\mathcal{V}_i$ sends his/her encrypted vote to the Blockchain as well as a proof that the submitted score falls within a publicly defined range $[0, P]$. To do this, we use a ZKP to prove that this condition is met. In order to accomplish this, we utilize a zero-knowledge set membership proof for each score to show that the score falls within the range without disclosing the assigned score.

Before submitting the vote to the Blockchain, the voter first authenticates each element in ϕ by using the election authority's public key. The main goal of this phase is to allow voters to verify that their votes will be submitted to the elections they belong to. This verification is useful especially when elections are held on different states and each state has a corresponding election authority's public key. In that case, the voter will verify that his vote is added to the state where he belongs. The verification is done by checking the following equality [34]:

$$e(\mathcal{A}_i, y \cdot g^{p_i}) \stackrel{?}{=} e(g, g)$$

The voter needs to prove in zero-knowledge that the score he/she selected lies in the set, as follow:

1. The voter picks $v \in_R \mathbb{Z}_p$ and computes $V = \left(\mathcal{A}_{p_o^{(r)}}\right)^v$, where $A_{p_o^{(r)}}$ is the election authority' signature on $p_o^{(r)}$;
2. The voter picks random numbers $t, m \in_R \mathbb{Z}_p$ and computes

$$a = e(V, g)^{-s} e(g, g)^t, \text{and } \mathcal{Q} = g^s h^m.$$

However, the underlying ZKSM [35] needs an interactive session between both the prover and the verifier and thus will lead to extra communication overheads due to multiple rounds of communication. Our solution is to leverage the Fiat–Shamir heuristic [36], to convert the interactive zero-knowledge scheme to a non-interactive protocol. Using the Fiat–Shamir heuristic, a challenge c can be calculated from public parameters as follows

$$c = \mathcal{H}(\mathcal{M})$$

where the $(\mathcal{M})$ is the smart contract address that holds the election.
Then, he/she computes $z_p = s - \left(p_o^{(r)}\right)c$, $z_v = t - vc$ and $z_\iota = m - \iota c$. Then, the proof of ZKSM is denoted as $\pi = C \| c \| a \| \mathcal{Q} \| z_p \| z_v \| z_\iota$.
Then, the voter needs to send a vote for each candidate by using the committed value and masks it with the shared secret $\mathcal{S}_{v_i}$ as follows.

$$\mathcal{V}_{v_i}^{c_j} = \mathcal{S}_{vi} - P^{c_j}.c \tag{1}$$

where c is the challenge, and P^{c_j} is the score value. For example, assuming an election with two candidates and a voter v_1, selecting a score of 2 for the first candidate and a score 4 for the second candidate, two votes will be submitted, as follows.

$$\mathcal{V}_{v_1}^{c_1} = \mathcal{S}_{v1} - 2.c$$

$$\mathcal{V}_{v_1}^{c_1} = \mathcal{S}_{v1} - 4.c$$

Finally, he/she sends $\pi \| \mathcal{V}_{v_i}^{c_j}$ as a transaction to the Blockchain.

3. Once the contract $\mathcal{T}$ receives the proof π, the Blockchain checks if it is from the registered voters, and the time of receiving the proof lies in the election interval window. The proof that C is a commitment to an element in ϕ can be validated by the following statement [35].

$$\mathbf{PK}\left\{(p_o^{(r)}, \iota, v) : C = g^{p_o^{(r)}} h^\iota \wedge V = g^{\overline{x + p_o^{(r)}}^v}\right\} \tag{2}$$

Proving that the statement in (2) is done in the Blockchain by checking the following conditions:

$$\mathcal{Q} \stackrel{?}{=} C^c h^{z_t} g^{z_p} \tag{3}$$

$$a \stackrel{?}{=} e(V,y)^c \cdot e(V,g)^{-z_p} \cdot e(g,g)^{z_v} \tag{4}$$

Once the two conditions in 3 and 4 are verified by the Blockchain, the vote will be added permanently to the Blockchain. The Blockchain will accept the authorized voters' vote if it is correctly constructed by the driver Due to the *Soundness* and *Completeness* property of ZKSM [35].

5.3. Tally Phase

The goal of this phase is to get the total results for the elections. Once all the candidate have submitted their votes and to get the summation of the scores for each candidate, anyone including the election authority can get the total votes per candidate as follows:

$$\sum \mathcal{V}_{v_i}^{c_j} = \sum \mathcal{S}_{vi} - \sum P^{c_j}.c$$

However, the $\sum \mathcal{S}_{v_i} = 0$, thus by dividing the $\sum \mathcal{V}_{v_i}^{c_j}$ by c, the total score for a candidate can be obtained.

6. Performance Analysis

The following section discusses the testing and evaluation of the proposed protocol on the Ethereum Blockchain. We first describe the configuration of the testing environment, Ethereum and Caliper, and define our key metrics for evaluating the performance of our protocol. Finally, we discuss the results of the experiments.

6.1. Methodology/Experiment Setup

The testing environment is configured in an instance of Linux Ubuntu 64-bit v20.04.3 LTS within Oracle VM Virtual Box, hosted on a Lenovo IdeaPad Flex 5 running Windows 11 Home with a 2.40 GHz 11th Generation Intel Core i5 processor.

Ethereum Setup. The system's smart contract is written in Solidity and the migration file is written in JavaScript. The system is deployed to a local instance of a Blockchain using Truffle. Truffle enables interactions with the smart contract and Blockchain by setting up a Blockchain instance and Ethereum network using Ganache, and is used for compiling and deploying the contract to the local Ganache Blockchain.

Hyperledger Caliper Setup. Hyperledger Caliper v0.4.2 is used to test the performance of our protocol on the Ethereum Blockchain network [37]. A Caliper, a Blockchain performance framework, is used to run tests on our protocol by using custom use cases that are defined by benchmark configurations and workload modules. When executed, Caliper produces standardized results that measure various metrics of the protocol's performance. These metrics are discussed in detail later. The benchmark configurations are used to define and structure the execution of the workload modules, as well as the collection of the test results. Each workload module defines the construction of a transaction and its submission to the Blockchain. Finally, a network configuration file binds the Caliper tool to our Ethereum network hosting our smart contract. The network is referred to as the SUT (System Under Test). In our implementation, workloads were written for Register, Vote, and Verify. Each benchmark is broken into different rounds, with each round allowing for some type of variation from the previous round. In our case, each round executes a different workload module (i.e., a different transaction); as such, each round specifies the transaction to be executed, the total number of transactions to be submitted (txCount), and the rate at which the transactions are to be executed (txRate). By manipulating these two variables, we explore their impact on the overall performance of the Blockchain, while keeping other factors such as the environment stable. The caliper records numerous metrics while monitoring the transaction executions, including the number of successful and failed

transactions, the send rate (tps), minimum, maximum, and average latency (seconds), and throughput (tps). These metrics are written to a report that can be viewed both via console and web browser.

6.2. Performance Metrics and Simulation Designs

Several metrics are used to evaluate the performance of the e-voting scheme, in the following, we introduce these metrics.

- *Throughput.* Throughput is the rate at which valid transactions are being added to the Blockchain. Throughput is measured in transactions per second (tps);
- *Latency.* Latency is the time between the issuance of a transaction and the receipt of a response from the Blockchain, i.e., the time it takes for the transaction data to become useful within the network. By measuring the average latency of a set of transactions, we are able to obtain an evaluation that accurately reflects the true latency that a user of the proposed protocol can expect;
- *Error Rate.* This metric is defined as the ratio of failed transactions to committed transactions. This figure indicates how much stress operation stress the Blockchain network is experiencing and how it should remain low to signify that the network is able to manage the transaction load.

To consider different cases where voters interact with the Blockchain, we measure the performance of the proposed protocol on the Blockchain network considering the following scenarios.

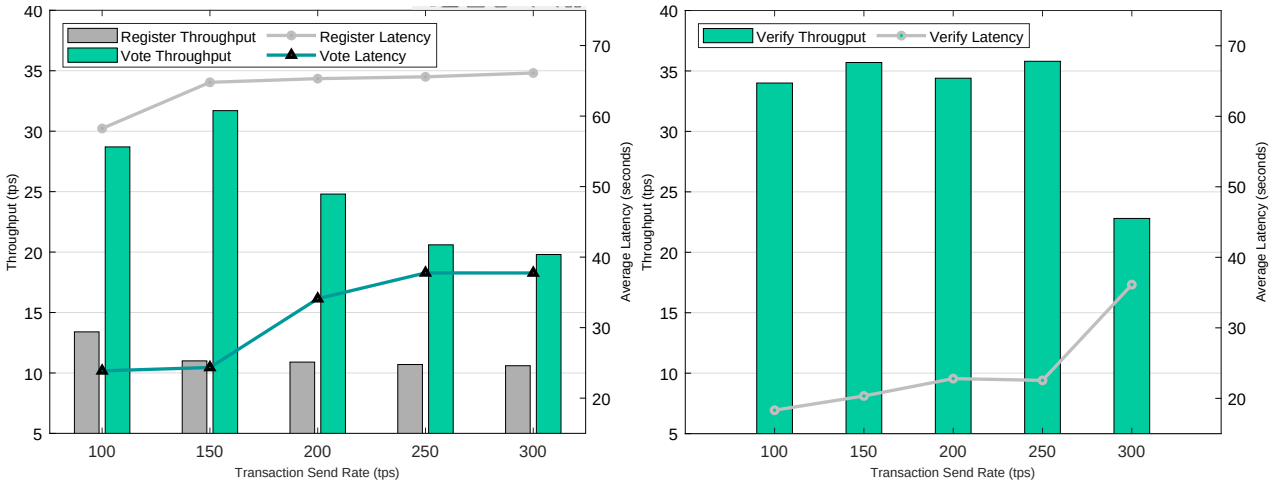
- *Case I.* This case executes a fixed number of transactions (1000 tx) while varying the transaction rate from 100 tps to 300 tps, and incrementing by a value of 50 tps. The goal of this case is to find an optimal upper bound for the transaction rate without producing significant negative impact on the network performance;
- *Case II and Case III.* These two cases bear the responsibility of investigating the impact of varying transactions with the ideal transaction rate determined in *Case I*. We test the total transaction amounts of 1000, 5000, 10,000, and 20,000 using a rate of 100 tps found in *Case I* with the purpose of discovering the impact increasing the transaction amounts on throughput and latency in *Case II*. *Case III* analyzes the impact on the error rate;
- *Case IV and Case V.* These two cases investigate the impact of varying transaction numbers and varying transaction rates on overall performance. Specifically, these tests can give an insight in the impact of varying the transaction rates on performance as the number of total transactions are increased. *Case IV* analyzes the impact on latency and throughput while *Case V* examines the impact on error rate.

6.3. Results

This section analyzes the results of the conducted experiments. For simplicity, the transactions are split into read transactions to represent transactions such as *tally*, which are used to query the Blockchain and write transactions to represent the *setup* and *Vote* algorithms.

Case I. The effect of increasing the transaction rates on the latency and throughputs is illustrated in Figure 3. It can be seen that the read operations boast considerably higher throughput than the write operations, and also modestly low latency. For example, at a send rate of 200 tps, the read operation has a latency of around 10 s whereas the latency is 16 and 35 s for the transactions that are writing to the Blockchain. Latency appears to bear a modest positive correlation to transaction send rate; the trend seems to be very modest and appears to become more steady after 200 tps (disregarding the read transaction at 300 tps as an outlier, likely due to a hardware slowdown). Throughput appears to decline as the transaction send rate increases, especially for the write transactions; the send rate seems to have less of a correlation with the throughput of read transactions. Additionally, throughput and latency appear to be negatively correlated, as many of the times a stark

jump in latency is observed when a corresponding drop in throughput is recorded, and vice versa. This leads uys to conclude the following: the transaction send rate does not have a significant influence throughput; rather, a strong negative correlation between latency and throughput is exhibited. *The transaction send rate of 200 tps was selected as the optimal send rate for the simulation of* Case II *and* Case III (at this benchmark the observed send rate was closest to the stipulated send rate of 200 tps).



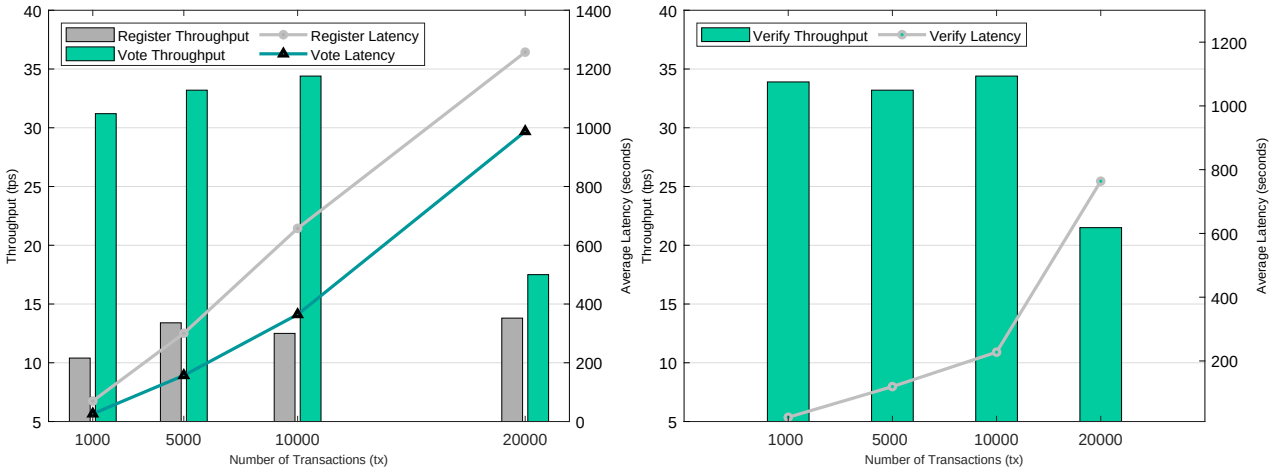
(**a**) Write Transactions (**b**) Read Transactions

Figure 3. Impact of increasing transaction rates on latency and throughput.

Case II and Case III. Figure 4 illustrates how increasing the number of transactions affects the throughput and latency of the SUT at a fixed transaction rate of 200 tps. It becomes immediately obvious that latency significantly increases after 10,000 transactions for both read and write transactions. Likewise, throughput experienced a significant decline after 10,000 transactions after remaining relatively constants for all transactions. It is interesting to note that despite a significant increase in latency at 20,000 transactions, there is a slight decline in throughput; in fact, it appears to exhibit a slight upward trend, signifying that it seems to increase as the number of transactions sent increases. This could be attributed to the fact that the latency increases more linearly whereas it increases more exponentially for the other transactions. In other words, it experiences less of a latency increase relative to the increase that the other transactions exhibit and thus shows less of a throughput decline relative to the other functions. This would then be expected given the correlation between latency and throughput. The error rate for transactions submitted at 200 tps remained at 0% regardless of the number of transactions, and remained at 0% for other transaction send rates, as well. It is plausible to believe that this perfect success rate on the Ethereum network negatively impacts the performance of the SUT and may be partially responsible for the exponential increases in latency. In summary:

- At a rate of 200 transactions per second, the throughput experiences little change up to 10,000 transactions. Beyond that, the throughput generally declines;
- Latency exponentially increases for read transactions. Latency increases in a more linearly fashion for write transactions;
- Likely in a trade-off with performance, Ethereum is able to execute the proposed scheme with a 0% error rate regardless of transaction send rates or the number of transactions.

To conclude, the proposed system is able to effectively handle transaction loads up to 10,000 at a rate of 200 tps.



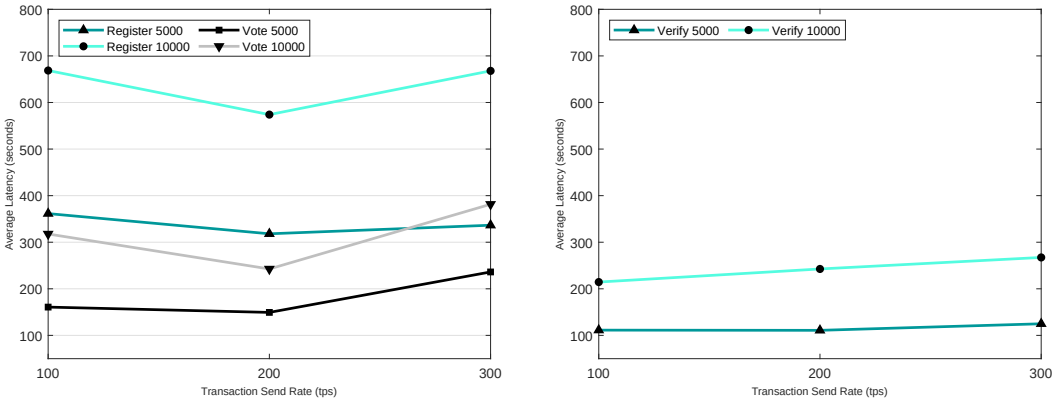
(**a**) Write Transactions (**b**) Read Transactions

Figure 4. The impact of increasing the number of transactions on latency and throughput.

Case IV and Case V. The impact of increasing both the transaction send rate and the number of transactions on latency and throughput can be seen in Figures 5 and 6, respectively. Transactions of 5000 and 10,000 are each executed at 100, 200, and 300 tps. It is clear that the read transactions experienced the lowest latency; additionally, it can be noticed that in the case of 5000 transactions, increasing the transaction send rate has no significant effect on the latency while in the case of 10,000 transaction, there is only a slight increase in latency. The results for the write transactions were more diverse. Increasing the number of transactions increases the latency more significantly for *Register* than for *Vote* (by a factor of roughly 3); and the transaction rate had little discernible effect on the throughput of the write transactions. It is interesting to note that across all three of the four charts, the latency and throughput were best at 200 tps. This supports our earlier conclusion that 200 tps was the best send rate with which to conduct the simulation of *Case II*. Much as before, the error rate for transactions remained at 0% despite varying both the transaction send rate and the number of transactions. To summarize:

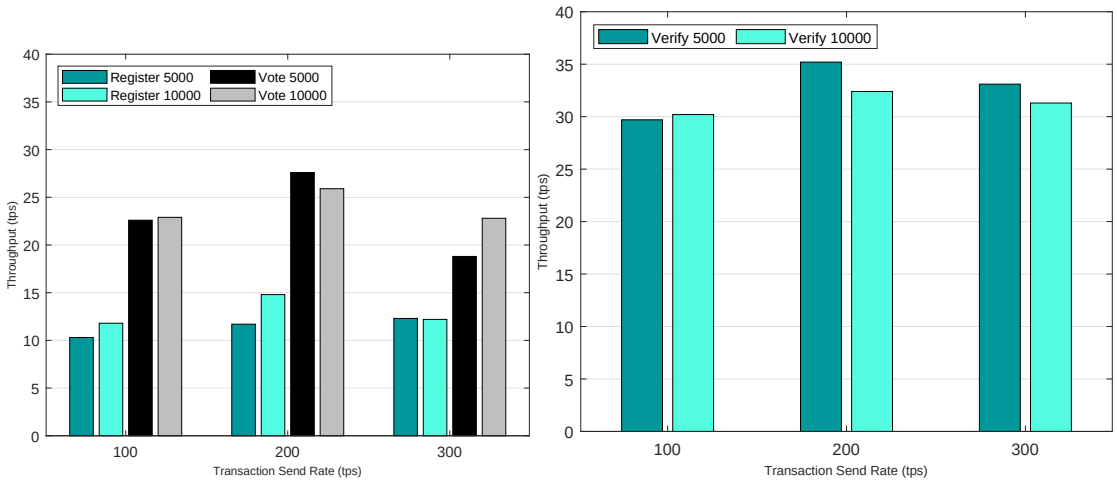
- Varying the transaction send rates only has a marginal negative effect on the latency for read transactions while it includes a positive effect on the throughput for read transactions, most notably at a send rate of 200 tps;
- Increasing the transaction send rate (up to 200 tps) has no significant impact on the throughput and latency of the read and write transactions;
- A transaction send rate of 200 tps produces the highest performance from the SUT and regardless of the number of transactions or their send rate, Ethereum is able to maintain a 0% error rate while executing the proposed scheme.

In conclusion, the proposed system is feasible with a transaction workload of up to 10,000 transactions before its performance starts to become compromised. This performance is maximized if the transactions are sent at a rate of 200 tps.



(**a**) `Write Transactions` (**b**) `Read Transactions`

Figure 5. Impact of increasing the transaction rate and number of transactions on latency.



(**a**) Write Transactions (**b**) Read Transactions

Figure 6. The impact of increasing the number of transactions on latency and throughput.

7. Security and Privacy Analysis

In this section, we will discuss the security and privacy concerns and how the proposed approach mitigates them.

Preposition 1. *Our proposed scheme preserves the privacy of voters by masking their votes.*

Proof. A voter in the proposed scheme interacts with a randomly generated *one-request-only* Blockchain address, i.e., the corresponding public key. Then, the voter shall prove their knowledge to the public key in a zero-knowledge before submitting their votes to the Blockchain. The proposed scheme preserves the voters' privacy by masking their votes before it is submitted to the Blockchain. □

Preposition 2. *The proposed scheme ensures election fairness by mitigating malicious voters who may manipulate the voting scores.*

Proof. The yes/no voting approach requires an eligible voter to submit a yes/no for a candidate, while the score voting requires the voter to submit a score in a predefined range for each candidate. To preserve privacy, votes shall be encrypted before it is submitted to the Blockchain, however, a malicious voter may falsely submit a score that is not in the range to manipulate the election result in the favor of a certain candidate. The proposed approach ensures the security against malicious behavior of these voters by leveraging the zero-knowledge set membership proof such that the voter shall prove first in a zero-knowledge that the score selected lies in the predefined set without allowing the Blockchain

and/or eavesdroppers to know the score value per candidate. In the proposed scheme, the Blockchain will accept the authorized voters' vote if and only if the voter has correctly selected a score value that is in the set defined by the election authority. This is due to the *Soundness* and *Completeness* properties of ZKSM [35]. This ensures the fairness of the elections. □

Preposition 3. *Our proposed scheme does not suffer from a single point of failure.*

Proof. The underlying Blockchain is used to execute the proposed scheme and even if 51% of nodes are malicious, they cannot collude to obtain the election results because the designed scheme requires the knowledge of all masked values shared by the election authority. This is because the S is divided among all the voters so our tallying scheme is secure against the 51% attack. □

Preposition 4. *Passive eavesdroppers can not infer any private information about the Ballot secrecy.*

Proof. One of the threats is to protect the score voting scheme from passive adversaries who are not actively engaged in the process of voting, but instead are trying to gain knowledge of the election proceedings by eavesdropping on communication channels and the Blockchain. This is done by masking the votes as well as using the ZKSM to prove that the submitted score lies in the set in a zero-knowledge manner. Therefore, a passive adversary cannot infer any sensitive information about the elections. □

8. Conclusions

This paper proposes a privacy-preserving ledger-based e-voting system that supports score-based ranked voting to ensure that the system maintains maximum ballot secrecy, fairness, and self-tallying. The proposed scheme allows voters to cast their votes encrypted to the Blockchain while allowing the voters to prove that the selected score lies in the predefined range to ensure fairness. This verification is done in a zero-knowledge environment to preserve privacy. A set of experiments were set up to evaluate the performance of the proposed system, and an analysis of the results revealed that the current scheme is practical for both small and mid-sized elections where low latency is desired. Additionally, no matter the size of the workload, the Ethereum network is able to maintain a 0% error rate when executing the system's transactions, proving that, if deployed on the Ethereum network, our system has extremely high reliability. The proposed scheme is able to resolve the issue of maintaining the privacy of the voter and the election results while keeping the Blockchain transparent and verifiable through the use of zero-knowledge proofs.

Author Contributions: Conceptualization, A.A., G.S. and M.B.; formal analysis; funding acquisition, A.A. and M.B.; investigation, M.B.; methodology; A.A., G.S., W.R. and M.A. (Majed Almusali) project administration, G.S. and; resources, G.S. and M.A. (Majed Almusali); software, M.B.; supervision, G.S., M.B. and G.S.; validation, W.R. and M.A. (Majed Alrowaily); writing—review and editing, A.A., G.S., W.R., M.A. (Majed Alrowaily), M.A. (Majed Almusali) and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at University of Tabuk for funding this work through Research No. 0270-1443-S.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at the University of Tabuk for funding this work through Research no. 0270-1443-S.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. What is An Online Voting System? 2022. Available online: https://www.eballot.com/votes-and-elections/what-is-an-online-voting-system (accessed on 1 January 2023).
2. Dery, L.; Tassa, T.; Yanai, A.; Zamarin, A. A secure voting system for score based elections. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 2399–2401.
3. Li, Y.; Susilo, W.; Yang, G.; Yu, Y.; Liu, D.; Du, X.; Guizani, M. A blockchain-based self-tallying voting protocol in decentralized IoT. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 119–130. [CrossRef]
4. Could Estonia Be the Model for Secure Online Voting? 2022. Available online: https://www.govtech.com/blogs/lohrmann-on-cybersecurity/could-estonia-be-the-model-for-secure-online-voting.html (accessed on 1 January 2023).
5. Internet Voting in Estonia. 2022. Available online: https://www.ndi.org/e-voting-guide/examples/internet-voting-in-estonia (accessed on 1 January 2023).
6. Safi, M.; Chan, G. NSW Election Result Could be Challenged over iVote Security Flaw. 2015. Available online: https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw (accessed on 1 January 2023) .
7. Internet or Online Voting Remains Insecure. 2022. Available online: https://www.aaas.org/epi-center/internet-online-voting (accessed on 1 January 2023).
8. Specter, M.A.; Koppel, J.; Weitzner, D. The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in {US}. Federal Elections. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Baltimore, MD, USA, 12–14 August 2020; pp. 1535–1553.
9. Badr, M.; Baza, M.; Sherif, A.; Mahmoud, M. Blockchain-Based Ride-Sharing System with Accurate Matching and Privacy-Preservation. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021.
10. Badr, M.M.; Ibrahem, M.I.; Baza, M.; Mahmoud, M.; Alasmary, W. Detecting Electricity Fraud in the Net-Metering System Using Deep Learning. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–6.
11. Abdelfattah, S.; Baza, M.; Badr, M.M.; Mahmoud, M.M.; Srivastava, G.; Alsolami, F.; Ali, A.M. Efficient Search Over Encrypted Medical Data With Known-Plaintext/Background Models and Unlinkability. *IEEE Access* **2021**, *9*, 151129–151141. [CrossRef]
12. Onur, C.; Yurdakul, A. ElectAnon: A Blockchain-Based, Anonymous, Robust and Scalable Ranked-Choice Voting Protocol. *arXiv* **2022**, arXiv:2204.00057.
13. Yang, Y.; Guan, Z.; Wan, Z.; Weng, J.; Pang, H.H.; Deng, R.H. PriScore: Blockchain-Based Self-Tallying Election System Supporting Score Voting. *IEEE Trans. Inf. For. Secur.* **2021**, *16*, 4705–4720. [CrossRef]
14. Zeng, G.; He, M.; Yiu, S.M.; Huang, Z. A Self-Tallying Electronic Voting Based on Blockchain. *Comput. J.* **2021**, *64*, 3020–3034. [CrossRef]
15. Panja, S.; Roy, B. A secure end-to-end verifiable e-voting system using blockchain and cloud server. *J. Inf. Secur. Appl.* **2021**, *59*, 102815. [CrossRef]
16. Li, H.; Li, Y.; Yu, Y.; Wang, B.; Chen, K. A blockchain-based traceable self-tallying E-voting protocol in AI era. *IEEE Trans. Netw. Sci. Eng.* **2020**, *8*, 1019–1032. [CrossRef]
17. Clarkson, M.R.; Chong, S.; Myers, A.C. Civitas: Toward a secure voting system. In Proceedings of the 2008 IEEE Symposium on Security and Privacy (sp 2008), IEEE, Oakland, CA, USA, 18–21 May 2008; pp. 354–368.
18. Sandler, D.; Derr, K.; Wallach, D.S. VoteBox: A Tamper-evident, Verifiable Electronic Voting System. In Proceedings of the USENIX Security Symposium, Santa Clara, CA, USA, 14 August 2008; Volume 4, p. 87.
19. Voatz Mobile Voting App Deemed Insecure by MIT Researchers. 2020. Available online: https://www.techtarget.com/searchsecurity/news/252478605/Voatz-mobile-voting-app-deemed-insecure-by-MIT-researchers (accessed on 1 January 2023).
20. Han, G.; Li, Y.; Yu, Y.; Choo, K.K.R.; Guizani, N. Blockchain-Based Self-Tallying Voting System with Software Updates in Decentralized IoT. *IEEE Netw.* **2020**, *34*, 166–172. [CrossRef]
21. Dery, L.; Tassa, T.; Yanai, A. Fear not, vote truthfully: Secure Multiparty Computation of score based rules. *Expert Syst. Appl.* **2021**, *168*, 114434. [CrossRef]
22. Panja, S.; Bag, S.; Hao, F.; Roy, B. A Smart Contract System for Decentralized Borda Count Voting. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1323–1339. [CrossRef]
23. Yi, H. Securing e-voting based on blockchain in P2P network. *Eurasip J. Wirel. Commun. Netw.* **2019**, *2019*, 137. [CrossRef]
24. Utah Green Party Hosts Dr. Stein; Elects New Officers. 2017. Available online: https://independentpoliticalreport.com/2017/06/utah-green-party-hosts-dr-stein-elects-new-officers/ (accessed on 1 January 2023).
25. Score Voting. 2022. Available online: https://electionscience.org/library/score-voting/ (accessed on 1 January 2023).
26. Baza, M.; Lasla, N.; Mahmoud, M.M.E.A.; Srivastava, G.; Abdallah, M. B-Ride: Ride Sharing with Privacy-Preservation, Trust and Fair Payment Atop Public Blockchain. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 1214–1229. [CrossRef]
27. What is Blockchain Technology? 2022. Available online: https://www.ibm.com/topics/what-is-blockchain (accessed on 1 January 2023).

28. Bhutta, M.N.M.; Khwaja, A.A.; Nadeem, A.; Ahmad, H.F.; Khan, M.K.; Hanif, M.A.; Song, H.; Alshamari, M.; Cao, Y. A survey on blockchain technology: Evolution, architecture and security. *IEEE Access* **2021**, *9*, 61048–61073. [CrossRef]
29. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasubramanian, V. A survey on the adoption of blockchain in iot: Challenges and solutions. *Blockchain Res. Appl.* **2021**, *2*, 100006. [CrossRef]
30. Ali, O.; Jaradat, A.; Kulakli, A.; Abuhalimeh, A. A comparative study: Blockchain technology utilization benefits, challenges and functionalities. *IEEE Access* **2021**, *9*, 12730–12749. [CrossRef]
31. Camenisch, J.; Stadler, M. Efficient group signature schemes for large groups. In Proceedings of the Advances in Cryptology—CRYPTO '97, Santa Barbara, CA, USA, 17–21 August 1997; Kaliski, B.S., Ed.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 410–424.
32. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 839–858.
33. McCorry, P.; Shahandashti, S.F.; Hao, F. A smart contract for boardroom voting with maximum voter privacy. In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017; Springer: Berlin, Germany, 2017; pp. 357–375.
34. Boneh, D.; Boyen, X. Short signatures without random oracles. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 56–73.
35. Camenisch, J.; Chaabouni, R.; Shelat, A. Efficient protocols for set membership and range proofs. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, 7–11 December 2008; Springer: Berlin, Germany, 2008; pp. 234–252.
36. Fiat, A.; Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, Amsterdam, The Netherlands, 10 May 1986; pp. 186–194.
37. Hyperledger Caliper v0.4.2. 2022. Available online: https://hyperledger.github.io/caliper/v0.4.2/architecture/ (accessed on 1 January 2023).