An Efficient Open Vote Network for Multiple Candidates

SANGSOO SEOL\textsuperscript{1}, HYOSEUNG KIM\textsuperscript{1}, AND JONG HWAN PARK\textsuperscript{2}

\textsuperscript{1}Graduate School of Information Security, Korea University, Seoul 02841, South Korea
\textsuperscript{2}Department of Computer Science, Sangmyung University, Seoul 03016, South Korea
Corresponding author: Jong Hwan Park (jhpark@smu.ac.kr)

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00518, Blockchain privacy preserving techniques based on data encryption).

\section*{ABSTRACT}
In 2010, Hao et al. proposed an efficient decentralized voting protocol known as Open Vote Network (OV-Net), which does not require a trusted third party to count the votes or a private channel to protect ballot secrecy. In the last decade, various studies have been conducted to solve some limitations of the OV-Net, such as fairness and robustness. However, an unresolved problem exists in the OV-Net (and its variants), which is its limited scalability for multiple candidates. The computational cost for tallying, which increases exponentially with the number of candidates, causes the scalability problem. Therefore, in this study, we solve this problem by proposing a variant of the OV-Net for multiple candidates and showing that the computational cost of tallying increases linearly with the number of candidates. Regarding security, we prove that our variant satisfies the ballot secrecy and dispute-freeness in formal security models, based on the decision Diffie-Hellman assumption and non-interactive zero-knowledge properties, respectively. Moreover, regarding the efficiency, we compare the performance of the traditional and proposed OV-Net from the theoretical standpoint and present experimental results to show that the proposed OV-Net variant is considerably more efficient than the traditional OV-Net as the numbers of voters and candidates increase.

\section*{INDEX TERMS}
Open vote network, self-tallying, scalability, ballot secrecy, dispute-freeness, one-man-one-vote rule.

\section*{I. INTRODUCTION}
Election plays a crucial role in modern society, and electronic voting is one of the most intriguing research topics in the field of cryptography. Several studies have been proposed for building electronic voting systems with the development of internet technology as an alternative for solving numerous limitations of paper-based elections.

Since Chaum [1] proposed the first electronic voting system using his mix network protocol in 1981, subsequent voting protocols [2], [3], [4] have been constructed, which typically depend on a trusted third party to protect the privacy of voters. Interestingly, in these centralized constructions, a trusted third party collects encrypted votes for all voters and computes a final tally from the encrypted votes.

However, a fatal flaw of the centralized voting protocol is that a trusted third party becomes the single point of failure while performing the entire voting protocol. Moreover, all voters must believe that the trusted third party, who fully controls the voting protocols, behaves honestly. To alleviate this trust issue from centralization, threshold cryptography can be considered to distribute this strong trust across multiple authorities [5]; however, voters must still trust that multiple authorities do not collude.

Additionally, several studies on building decentralized voting systems [6], [7], [8], where voters run a voting protocol without any trusted third party, have been conducted to resolve the trust issue. In 2002, Kiayias and Yung [6] first introduced a self-tallying voting protocol where no authority is required for tallying. Certainly, the tallying process in [6] can be conducted as an open procedure by allowing anyone (including non-voters) to compute tallying without any
external support, considering the encrypted votes published by all voters. In 2004, Groth [7] proposed a novel self-tallying voting protocol to improve the efficiency of the Kiayias and Yung protocol. In 2010, inspired by the anonymous veto network [9], Hao et al. [8] presented a two-round self-tallying voting protocol called Open Vote Network (OV-Net), which is considerably more efficient than [6] and [7] and as secure as both.

However, scalability is the main limitation of the OV-Net (including its predecessors [10], [11]), indicating that the performance of the OV-Net deteriorates rapidly as the number of voters or candidates increases. The scalability problem emanates from the inefficiency of self-tallying. Specifically, the self-tallying of the OV-Net operates with the “encrypt-then-cancel” paradigm, where each vote is first encrypted, all the randomness used for encryption is canceled during tallying calculation, and the sum of all votes is raised as a discrete logarithm of a group element. Consequently, for the self-tallying, it is necessary to determine the discrete logarithm via an exhaustive search. The problem is then generated in the possible set of exponents that the discrete logarithm can accommodate. For $n$ voters and $k$ candidates, the set of exponents in the OV-Net has approximately $O(n^k)$ elements that increase exponentially with respect to the number $k$; therefore, the relevant self-tallying needs to compute approximately $O(n^k)$ multiplications in a group. For example, if $n = 1024$ and $k = 5$, then the self-tallying must calculate about $2^{50}$ multiplications, which is a cumbersome computation for each voter.

One may think that instead of performing multiplications for exhaustively searching every tallying, a table of precomputed multiplications could be used to simply search for a value (in the precomputed table) corresponding to a discrete logarithm in question. In the OV-Net, however, this approach requires considerable storage capacity. When $n = 1024$ and $k = 5$, as in the above case, each voter who must self-tally in a decentralized environment performs approximately $2^{50}$ multiplications and stores the resulting values of approximately $2^{50}|G|$ in size, where $|G|$ is the size of a group element. Moreover, when the numbers of voters and candidates slightly increase, the storage capacity (including the computational cost) required to achieve the precomputed table will increase exponentially.

Despite the abovementioned limitations, the scalability issue remains unsolved in the OV-Net and its variants [10], [11], [12], [13], [14]. This means that they should work under the very constrained condition that the number of voters or candidates is small. One example is a “yes/no” voting option where the number of candidates is two. Furthermore, because voting for multiple candidates is frequently used in reality, it can be a crucial challenge to present an efficient voting system to solve the scalability issue. To the best of our knowledge, thus far, no such voting system for multiple candidates has been proposed in a decentralized environment.

A. OUR CONTRIBUTIONS

We propose a new variant of the OV-Net [8] for multiple candidates that is much more scalable than the traditional OV-Net, in a sense that the computational cost for tallying increases linearly with the number of candidates. Table 3 shows the performance comparison between the traditional OV-Net and ours, with an important difference in the tallying process. The table shows that for a fixed number $n$ of voters, the tallying computation increases exponentially in the traditional OV-Net, but increases polynomially in ours with increasing number $k$ of candidates. Therefore, we summarize our contribution as follows:

- The core idea of our construction is to first run the traditional OV-Net in parallel with two options (i.e., a yes/no vote) independently for $k$ candidates and then apply our novel OV-Net technique that guarantees a one-man-one-vote rule among the $k$ votes. With our novel technique, anyone (including non-voters) can publicly verify that each voter only has to vote for one of the $k$ candidates, which is another realization of the one-man-one-vote rule. In this process, each vote value generated by the two OV-Nets is bound using non-interactive zero-knowledge (NIZK) to prove the equality of discrete logarithms. Our proposed technique is shown in Figure 3, and our construction details are described in Section IV.

- We clarify the security model for ballot secrecy, which captures that an adversary gains no information about votes by honest voters. Our model considers a malicious adversary from the beginning, which is stronger than the prior model [10], where an adversary is assumed to be honest-but-curious. Additionally, we define a new security model for dispute-freeness, which captures that a malicious adversary cannot break the one-man-one-vote rule without being detected. Therefore, using these (new) security models, we prove that our construction satisfies the ballot secrecy and the dispute-freeness, respectively, relying on the decision Diffie-Hellman (DDH) assumption and NIZK properties. Our security proof is provided in Section V.

- To demonstrate the efficiency of ours in terms of scalability, we provide a performance comparison between the traditional and proposed OV-Net, based on specific experiments for measuring time. In our first experiment, the number of voters is fixed at 40 and the number of candidates is gradually increased to measure the time required for a voter to complete each voting protocol. Conversely, in our second experiment, the number of candidates is fixed at 3 and the number of voters is increased by 50. In both experiments, we can observe that ours is highly efficient compared with the traditional OV-Net when the number of voters or candidates increases, as expected from the theoretical difference presented in Table 3. The details of our experiments are given in Section VI-B.
B. RELATED WORK
In 2010, Hao et al. proposed the OV-Net, which can be used efficiently when two candidates exist (e.g., “yes/no” voting) in [8], along with three extended versions of the OV-Net that can be used in an environment with three or more candidates, and two of them (III-B2 and III-B3) guarantee the one-man-one-vote rule using 1-out-of-k zero-knowledge proofs [15] for the k candidates. However, one must perform $O(n^k)$ multiplication operations to tally in both extended versions. Due to the inefficiency of these operations, both extensions can only be used in environments with a small number of candidates.

The computational cost required for each voter’s participation in the OV-Net (including both extensions) is largely categorized into the “voting” and “tallying” phases. Additionally, the sum of the two phases is directly related to the scalability of the OV-Net for multiple candidates. In 2019, Lin et al. proposed an OV-Net variant [12] for multiple candidates to improve the scalability, using efficient 1-out-of-k zero-knowledge proofs employed in the voting phase. Although the computational cost in the voting phase has improved, the cost in tallying phase, which has the greatest effect on scalability, remains unresolved. Therefore, the computational cost required for tallying per voter in [12] increases exponentially with the number of candidates; thus, the assumption that the number of candidates is small is inevitable.

II. PRELIMINARIES
A. OV-Net
The OV-Net [8] is a two-round voting system that supports a relatively small number of voters, and does not require any trusted third party. The OV-Net comprises the following four phases:

- **Setup**: Given the numbers of voters and candidates, a list of all eligible voters and candidates is set up. All voters agree on all public system parameters.
- **Round 1**: Each voter generates public/private voting keys. Based on all the public voting keys published by all the voters, each voter computes public auxiliary values to be used to encrypt his/her own vote.
- **Round 2**: Each voter encrypts a vote using his/her private voting key and public auxiliary values computed in Round 1 and subsequently publishes the encrypted vote.
- **Tallying**: Given all encrypted votes published by all voters, anyone (including non-voters) can count the votes corresponding to each candidate.

B. DECISION DIFFIE-HELLMAN ASSUMPTION
1) THE DECISION DIFFIE-HELLMAN PROBLEM
Let $G$ be a group of prime order $p$ and $g$ be a generator of $G$. Given a tuple $(g, g^a, g^b, T)$ where $a, b \in \mathbb{Z}_p$, the decision Diffie-Hellman (DDH) problem [16] is to decide whether $T = g^{ab}$ in $G$.

We say that the DDH assumption holds in $G$ if it is infeasible for a probabilistic-polynomial-time adversary to solve a DDH problem in $G$.

C. NON-INTERACTIVE ZERO-KNOWLEDGE PROOF
A non-interactive zero-knowledge (NIZK) proof system [17], [18] for a language $L$ comprises the following algorithms (Setup, Prove, Verify) such that:

- **Setup**($1^\kappa$) takes a security parameter $\kappa$ and outputs a common reference string $\sigma$.
- **Prove**($\sigma, s, w$) takes the common reference string $\sigma$, a statement $s \in L$, and a witness $w$ as input and outputs a proof $\pi$.
- **Verify**($\sigma, s, \pi$) takes the common reference string $\sigma$, a statement $s \in L$, and a proof $\pi$ as the input and outputs 1 when it accepts the proof and 0 otherwise.

We consider a NIZK proof system that satisfies the following properties:

- **Completeness**: A verifier should successfully verify the proof generated by an honest prover who has a witness.
- **Soundness**: For a false statement $s$, a cheating prover cannot falsely convince the honest verifier that $s \in L$.
- **Zero-knowledge**: A simulator $S$ exists, not knowing a witness, which can generate a simulated proof indistinguishable from a real one.

Instead of soundness, we often consider the following NIZK property known as extractability:

- **Extractability**: Whenever a (potentially cheating) prover produces two valid proofs with respect to a statement, an efficient extractor based on the forking lemma [19] and random oracles can extract the witness from the information available to the adversary.

D. NIZK PROPERTIES FOR OUR SECURITY PROOFS
Our construction uses three NIZK proof systems of NIZK$_1$, NIZK$_2$, and NIZK$_3$, as described in Figures 1, 2, and 4, respectively. These NIZK proofs must satisfy the zero-knowledge property for our security proofs, which can be easily shown when $H$ is modeled as a random oracle. Next, NIZK$_1$ has the extractability property as a proof of knowledge and according to the slightly extended proof of equality of discrete logarithms [20] (cf. Lemma 1), NIZK$_3$ has the soundness property. We now show that NIZK$_2$ fulfills the soundness property using a proof similar to the proof presented in [20].

Claim 1: Assume that $v = 0$ or 1, and there is no $x$ such that $X = g^x$ and $U = Y^x g^s$. Then, for any $a_1$, $b_1$, $a_2$, and $b_2$ (described in Figure 2), there is at most one value $c$, for which the Verify algorithm returns 1.

Proof: Let $a_1$, $b_1$, $a_2$, and $b_2$ be the values such that the prover can output correct proofs $(s_1, s_2, d_1, d_2)$ and $(s_1', s_2', d_1', d_2')$ to two different hash values, $c$ and $c'$. As $d_1 + d_2 = c$ and $d_1' + d_2' = c'$, the inequality of $c \neq c'$ implies that either $d_1 \neq d_1'$ or $d_2 \neq d_2'$ must occur because otherwise $c = c'$.

In case when $d_1 = d_1'$, we observe that $g^{a_1} X^{d_1} = g^{a_1'} X^{d_1'}$ and $Y^{s_1} U^{d_1} = Y^{s_1'} U^{d_1'}$, resulting in the case of $X = g^{(s_1 - s_1')/(d_1' - d_1)}$, which contradicts the assumption of the claim. Similarly, in the case where $d_2 \neq d_2'$, we observe that $g^{a_2} X^{d_2} = g^{a_2'} X^{d_2'}$ and $Y^{s_2} U^{d_2} = Y^{s_2'} U^{d_2'}$, which contradicts the assumption of the claim.
\[ Y^{x_2} (U/g)^{d_2} = Y^{x_2} (U/g)^{d_2}, \] resulting in the case of \( X = g^{(x_2 - x_2')/(d_1 - d_2)} \) and \( U = Y^{(x_2 - x_2')/(d_1 - d_2)} g \), which are also a contradiction to the assumptions of the claim. \( \square \)

**III. PREVIOUS WORK**

**A. OV-NET**

In this subsection, we review the previous OV-Net [8] protocol, which enables a voter to cast only a “yes/no” vote. Next, we introduce several methods to extend [8] for multiple candidates voting protocols, which have scalability issues from our viewpoint. The OV-Net operates as follows:

1. **SETUP**

Let \( n \) be the number of voters and \((P_1, P_2, \ldots, P_n)\) be a list of all eligible voters. We assume that all voters agree on \((G, g)\), where \( G \) is a DDH-hard group of prime order \( p \) and \( g \) is a generator in \( G \).

2. **ROUND 1**

1. Each voter \( P_i \) selects a random exponent \( x_i \in_R \mathbb{Z}_p \) (so-called “private voting key”).
2. \( P_i \) computes his/her public voting key \( X_i = g^{x_i} \) using his/her private voting key.
3. \( P_i \) selects a random exponent \( x_i \), and the relevant \( \pi_{1,i} \).
4. \( P_i \) publishes his/her public voting key, \( X_i \), and the relevant \( \pi_{1,i} \).
5. In the end, \( P_i \) verifies all the NIZK proofs (related to \( \pi_{1,i} \)) published by all the voters. When all the proofs are valid, each voter \( P_i \) computes \( Y_i = g^{y_i} \), as given below:

\[
Y_i = \begin{cases} 
\frac{1}{\prod_{j=2}^{n} X_j} & (i = 1), \\
\prod_{j=1}^{i-1} X_j / \prod_{j=i+1}^{n} X_j & (2 \leq i \leq n - 1), \\
\prod_{j=1}^{n-1} X_j & (i = n).
\end{cases}
\]

3. **ROUND 2**

1. Each voter \( P_i \) chooses \( v_i \), which is either 0 or 1 vote (with respect to “yes” or “no”, respectively), and computes \( U_i = (Y_i)^{v_i} g^{v_i} \).
2. \( P_i \) generates \( \pi_{2,i} \) as a proof of NIZK2\{\((x_i, v_i) : X_i = g^{x_i} \land U_i = (Y_i)^{v_i} g^{v_i} \land v_i \in \{0, 1\}\}\) described in Figure 2.
3. \( P_i \) publishes \( U_i \) and the relevant \( \pi_{2,i} \).
4. In the end, \( P_i \) verifies all the NIZK2 proofs (related to \( \pi_{2,i} \)) published by all the voters. When all the proofs are valid, \( P_i \) proceeds to the tallying phase described below.

We note that \( \pi_{2,i} \) includes a 1-out-of-2 NIZK proof for proving that \( v_i \) is either 0 or 1 without revealing the witness \( v_i \), which was a non-interactive version of Cramer, Damgard, and Schoenmakers (CDS) [15] (see also [21]).

4. **TALLYING**

1. Each voter \( P_i \) computes \( \prod_{i=1}^{n} U_i = g^{\sum_{i=1}^{n} v_i} \).

\[
\text{NIZK}_1 \{ x : X = g^{x} \} 
\]

**Prove** \((x)\):

1. \( r \in_R \mathbb{Z}_p \)
2. \( a = g^r \)
3. \( c = H(i, g, X, a) \)
4. \( s = r - xc \)
5. Return \( \pi_2 = (c, s) \)

**Verify** \((X, \pi_1)\):

1. Parse \( \pi_1 \) as \((c, s)\)
2. \( a = g^s X^c \)
3. Check \( c = H(i, g, X, a) \)
4. Return 1 if the check passes; otherwise, return 0

**FIGURE 1.** NIZK1 proof system.

2. \( P_i \) calculates \( Y \) such that \( g^Y = \sum_{i=1}^{n} v_i \) by solving the discrete logarithm problem with respect to the base \( g \).

5. **CORRECTNESS**

In Round 1, we observe that

\[
y_i = \begin{cases} 
-x_i \sum_{j=2}^{n} x_j & (i = 1), \\
\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j & (2 \leq i \leq n - 1), \\
\sum_{j=1}^{n} x_j & (i = n).
\end{cases}
\]

In the Round 2, the exponential part of \((Y_i)^{v_i}\) becomes

\[
y_i x_i = \begin{cases} 
x_i \sum_{j=2}^{n} x_j & (i = 1), \\
x_i (\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j) & (2 \leq i \leq n - 1), \\
x_i \sum_{j=1}^{n} x_j & (i = n).
\end{cases}
\]

For ease of understanding, the sign of the \( x_i x_j \) value is provided in Table 1 when \( n = 5 \). The important point is that the sum of these \( x_i x_j \) values is zero, indicating that \( \prod_{i=1}^{n} (Y_i)^{v_i} = g^0 \). Because \( U_i = (Y_i)^{v_i} g^{v_i} \), we observe that, in the tallying phase, the following equation holds:

\[
\prod_{i=1}^{n} U_i = \prod_{i=1}^{n} g^{x_i x_i g^{v_i}} = g^{\sum_{i=1}^{n} v_i}
\]

Notably, the term \( \sum_{i=1}^{n} v_i \) is the number of “yes” votes, and the discrete logarithm of \( g^{\sum_{i=1}^{n} v_i} \) can be computed by exhaustive search, because \( \sum_{i=1}^{n} v_i \) is usually a small number bounded by \( n \) (the total number of voters).

**B. EXTENSION TO MULTIPLE VOTING OPTIONS**

When there are only two candidates, an election with two options (“yes” or “no”) described above can be used by
TABLE 1: Simple illustration of the sign of $x_i x_j$ for $n = 5$.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|-------|-------|-------|-------|-------|
| $x_1$ | -     | -     | -     | -     |
| $x_2$ | +     | -     | -     | -     |
| $x_3$ | +     | +     | -     | -     |
| $x_4$ | +     | +     | -     | +     |
| $x_5$ | +     | +     | +     | -     |

Interpreting the “yes” vote as one candidate and the “no” vote as the other candidate. However, when there are three or more candidates, the above protocol in which each voter has only two options cannot be used. To cover such an election for multiple candidates, three ways were suggested in [8] to extend the OV-Net with two options to the one with multiple candidates.

1) FIRST SUGGESTION
The first suggestion is to run the OV-Net (with two options) $k$ times in parallel for $k$ candidates, respectively. Each voter casts a “yes” or “no” vote for each candidate, and the tallying process is also done independently for each candidate.

2) SECOND SUGGESTION
The second suggestion is to use the technique employed in [6], [7], [8], and [21], which is to assign each candidate to a distinct (independent) generator. Specifically, an $i$-th candidate gets a generator $g_i$; thus, for $k$ candidates, $(g_1, g_2, \ldots, g_k)$ are assigned to each candidate. Round 1 is the same as that in the OV-Net, but in Round 2 each voter publishes $g^{v_i} g_i$ and 1-out-of-$k$ NIZK proof [15] for proving that $g_i \in \{g_1, g_2, \ldots, g_k\}$. For tallying, one computes

$$\prod_{i=1}^{n} g^{v_i} g_i = g_1^{v_1} g_2^{v_2} \cdots g_k^{v_k}$$

where $c_i$ is the count of votes for the $i$-th candidate. Then, one can determine each $c_i$ for all $i \in [k]$ using an exhaustive search, and the number of possible voting results is

$$(\frac{n+k-1}{k}) = O(n^{k-1})$$

by combination with repetitions [8].

3) THIRD SUGGESTION
The last suggestion is to use the property of super increasing number described in [22]. Let $n$ be the total number of voters. For the smallest integer $m$ such that $2^m > n$, each candidate is assigned to a power-of-2 integer ($2^m$) for proving that $v_i \in \{2^0, 2^m, \ldots, 2^{k-1}m\}$. To tally, one computes

$$\prod_{i=1}^{n} g^{v_i} g_i = g_1^{v_1} g_2^{v_2} \cdots g_k^{v_k}$$

and finds the value $\sum_{i=1}^{n} v_i$ via exhaustive search. Because $\sum_{i=1}^{n} v_i = 2^0 \cdot c_1 + 2^m \cdot c_2 + \cdots + 2^{k-1}m \cdot c_k$ where $c_i$ is the counts of votes for the $i$-th candidate and ($2^m$) has the super increasing nature, one can unambiguously compute the counts of votes for the $k$ candidates.

a: DISADVANTAGES
When the number of candidates is large, it is difficult to use the second and last suggestions because the computational cost for calculating the counts of votes for each candidate increases exponentially with the number of candidates. For instance, let $n = 100$ be the total number of voters and $k = 5$ be the number of candidates. Then, in the second suggestion, the computational cost for tallying is approximately $100^4 \approx 2^{26}$ trials. In the third suggestion, $m$ becomes 7 such that $2^m > 100$ and the tallying cost becomes approximately $(2^7)^k \approx 2^{35}$ trials when $c_k = 1$. When there is only one more candidate (in the case where $n = 100$ and $k = 6$), the second and third suggestions require the tallying cost to be approximately $2^{35}$ trials. Meanwhile, given the moderate number of voters and candidates, the first suggestion could be considerably scalable compared with the other two suggestions.
because the maximum trials for tallying is \( k \times n \). However, the first suggestion cannot be used in reality because of the critical issue that it cannot guarantee the realization of the one-man-one-vote rule, meaning that one voter can choose two or more candidates.

IV. OUR PROPOSED SYSTEM

A. IDEA

In this section, we propose an OV-Net variant that is much more efficient than aforementioned second and third suggestions when dealing with many candidates. The basic idea behind our construction is to develop a novel technique that ensures that (1) a vote for one candidate is 0 or 1 and (2) the sum of votes corresponding to all candidates becomes 1. Indeed, our technique can be viewed as one of the solutions to the one-man-one-vote rule, meaning that one voter can choose multiple candidates.

For proving the equality of discrete logarithms \( x_i \) for each candidate. Next, to ensure that \( \sum_{i=1}^{k} v_i = 1 \), that is, to prove the one-man-one-vote rule, meaning that one voter can choose one candidate independently. To achieve this, \( P_i \) publishes \( k \) public voting keys \( \{X_{i,1}, \ldots, X_{i,k}\} \), where \( X_{i,j} \) is the corresponding private voting key such that \( X_{i,j} = g^{v_{i,j}} \), where \( \{v_{i,1}, \ldots, v_{i,k}\} \) are raised by the same exponents, \( x_i \), and \( v_i \), for each candidate. Next, to ensure that the same vote was cast for each candidate, \( P_i \) is forced to generate a proof \( NIZK_F^k((x_{i,1}, v_{i,1}) : (g^{v_{i,2}}, x_{i,2}) \land \ldots \land (g^{v_{i,k}}, x_{i,k})) \) for proving the equality of discrete logarithms \( x_{i,j} \) and \( v_{i,j} \). Based on the soundness property of NIZK, we observe that \( (g^{v_{i,1}}, x_{i,1}) \) (used in Figure 3A) and \( (g^{v_{i,2}}, x_{i,2}) \) (used in Figure 3B) are raised by the same exponents, \( x_i \), and \( v_i \), for each candidate. Consequently, combining two OV-Nets (illustrated vertically and horizontally in Figure 3) with two relevant NIZK proofs allows us to efficiently realize the one-man-one-vote rule in the case of multiple candidates.

B. PROPOSED SYSTEM

Our proposed system architecture is depicted in Figure 5 and operates as follows:

1) SETUP

Let \( n \) and \( k \) be the numbers of voters and candidates, respectively. We use the notations \( (P_1, P_2, \ldots, P_n) \) and \( (C_1, C_2, \ldots, C_k) \) to represent all the voters and candidates, respectively. Therefore, we assume that all the voters agree on \((\mathbb{Z}, g)\).

2) ROUND 1

1. Each voter \( P_i \) selects \( k \) random secrets \( x_{i,j} \) for all \( i \in [k] \) as the private voting keys.
2. \( P_i \) computes \( X_{i,i} = g^{v_{i,i}} \) for all \( i \in [k] \) as the public voting keys.
3. \( P_i \) publishes his/her public voting keys along with \((\pi_{1,i}, \pi_{1,j}, \ldots, \pi_{1,k})\) where \( \pi_{1,i} \) is the proof of \( NIZK_F^k\{x_{i,j} : X_{i,j} = g^{v_{i,j}}\} \), which proves that \( P_i \) knows \( x_{i,j} \) corresponding to \( X_{i,j} \). Table 2 presents all public voting keys published by \( n \) voters.
4. In the end, \( P_i \) computes \( Z_{i,i} = g^{v_{i,i}} \) and \( Y_{i,i} = g^{v_{i,i}} \) for all \( i \in [k] \) as given below:

\[
Z_{i,i} = \prod_{j=2}^{k} X_{i,j} (i = 1),
\]

\[
\prod_{j=i+1}^{k} X_{i,j} (2 \leq i \leq k - 1),
\]

\[
Y_{i,i} = \prod_{j=2}^{k} X_{j,i} (l = 1),
\]

\[
\prod_{j=l+1}^{k} X_{j,i} (2 \leq l \leq n - 1),
\]

\[
\prod_{j=1}^{k} X_{j,i} (l = n).
\]

Note that two equations \( \prod_{i=1}^{k} (g^{v_{i,j}})^{x_{i,j}} = 1 \) and \( \prod_{i=1}^{k} (g^{v_{i,j}})^{x_{i,j}} = 1 \) hold. When using the public voting keys and their relevant private keys (Table 2), the OV-Net technique is applied vertically and horizontally, respectively.

3) ROUND 2

1. Each voter \( P_i \) sets \( v_{i,j} \) as either 0 (i.e., “no” vote) or 1 (i.e., “yes” vote) for all \( i \in [k] \), where only one of \( v_{i,j} \) is selected for each candidate. Consequently, combining two OV-Nets (illustrated vertically and horizontally in Figure 3) with two relevant NIZK proofs allows us to efficiently realize the one-man-one-vote rule in the case of multiple candidates.

4) TALLYING

1. Each voter \( P_i \) computes \( \prod_{j=1}^{k} U_{i,j} = g^{\sum_{j=1}^{k} v_{i,j}} \) for all \( i \in [k] \).
* i-th voter: \( P_i \)
* i-th candidate: \( C_i \)
* Encrypted vote for \( C_j \) published by \( P_i: (g^{u_{i,j}})^{x_{i,j}} \)

**FIGURE 3.** Sketch presenting the idea of our proposed system in case of three voters and three candidates.

**TABLE 2.** All public voting keys published from \( n \) voters to vote for \( k \) candidates.

| \( P_1 \) | \( g^{x_{1,1}} \) | \( g^{x_{1,2}} \) | … | \( g^{x_{1,k}} \) |
|\( P_2 \) | \( g^{x_{2,1}} \) | \( g^{x_{2,2}} \) | … | \( g^{x_{2,k}} \) |
|…|…|…|…|…|
|\( P_n \) | \( g^{x_{n,1}} \) | \( g^{x_{n,2}} \) | … | \( g^{x_{n,k}} \) |

* i-th voter: \( P_i \)
* Public voting key of \( P_i \) for j-th candidate: \( g^{x_{i,j}} \)

2. \( P_i \) calculates \( y_i \) such that \( g^{y_i} = \sum_{j=1}^{n} v_{i,j} \) by solving the discrete logarithm problem with respect to the base \( g \) for all \( i \in [k] \).

**Remark 1:** Using the same equation of \( \prod_{j=1}^{n} U_{j,i} = \prod_{j=1}^{n} g^{y_{j,i}} g^{v_{j,i}} \) employed in the OV-Net, \( y_i = \sum_{j=1}^{n} v_{i,j} \) is the sum of votes for i-th candidate \( C_i \). For each candidate, the number of tries for the exhaustive search is at most \( n \) for the total number of voters because of the one-man-one-vote rule. In addition, because \( \sum_{i=1}^{k} y_i = n \), the computational cost for exhaustive search is still \( n \) in the tallying phase.

**Remark 2:** An important advantage of our voting system is that it can easily be extended from 1-out-of-\( k \) to \( m \)-out-of-\( k \) voting, implying that each voter chooses \( m \) candidates among \( k \) candidates. The extension can be performed without any costly modification by simply changing the relation \( \prod_{i=1}^{k} V_{i,i} = g^{\sum_{i=1}^{k} v_{i,i}} = g \) (at the end of Round 2) into the one \( \prod_{i=1}^{k} V_{i,i} = g^{\sum_{i=1}^{k} v_{i,i}} = g^m \).

**Remark 3:** Furthermore, our voting system can consider the situation in which some voters cast their abstention votes. Specifically, when there are \( k \) candidates \( (C_1, C_2, \ldots, C_k) \), by adding one dummy candidate \( C_{k+1} \), anyone who wants to cast an abstention vote can vote for \( C_{k+1} \).

**FIGURE 4.** NIZK\(_3\) proof system.

**V. SECURITY ANALYSIS**

To analyze the security of our OV-Net variant, we consider two types of security requirements: ballot secrecy and dispute-freeness. Ballot secrecy assumes an active adversary who colludes with other corrupted voters to discover the secret ballot information of a target (honest) voter or tamper with voting results. In particular, when considering the ballot secrecy of a voting system, we must establish a security model while accepting two facts. First, we only consider "partial collusion" [8], which means that at least one honest voter except a target voter should exist (i.e., at least two voters should be honest) to consider an adversary for breaking the ballot secrecy of the target voter; otherwise, the adversary can easily get the ballot information of the target voter by corrupting all the other voters and performing...
FIGURE 5. Our proposed system architecture. Each order shown in the figure matches the order shown in IV-B.

A. BALLOT SECRECY

Hao et al. [8] provided an informal security analysis to show that the OV-Net fulfills ballot secrecy under the DDH assumption in 2010. Subsequently, Hao et al. [10] presented a formal security proof of ballot secrecy under a game-based framework where a challenger $C$ interacts with an adversary $A$. However, it is difficult to say that their security proof [10] fully reflected a realistic attack because (1) $A$ is assumed to be honest-but-curious (not malicious) and (2) both the zero-knowledge and soundness properties of NIZK (essential for OV-Net construction) are not analyzed. Because $A$ is assumed to behave honestly at the beginning of the protocol, $A$ can corrupt voters by issuing private key queries to $C$. $C$ knows all the private keys to answer such private key queries. Therefore, relying on those NIZK properties appears unnecessary, but one problem exists in proving ballot secrecy under the DDH assumption. Notably, for two target voters whose private keys are informally set to be discrete logarithms from an instance of the DDH problem, the zero-knowledge
property of NIZK is necessary for generating simulated NIZK proofs for these unknown discrete logarithms. Hence, the zero-knowledge property should not be dropped although \( A \) is considered honest-but-curious voters.

Considering the problems presented in [10], we define a new security model for ballot secrecy. The main difference between [10] and our new model is that \( A \) is considered malicious at the beginning of the protocol; therefore, \( A \) generates all the private keys for corrupted voters as per its choice without querying \( C \). For two target voters and one candidate that \( A \) chooses, \( A \) tries to decide which of the two voters will vote for the target candidate. As mentioned above, selecting two target voters fits the partial collision rule. Furthermore, having only one of the two vote for the target candidate is crucial to prevent \( A \) from trivially winning the ballot secrecy game from tallying results. Notably, our new model is stronger than that of [10] because the ballot secrecy against the malicious adversary implies that against the honest-but-curious adversary. Additionally, compared with [10], the ballot secrecy proof of our voting system will be analyzed by fully considering NIZK properties. We define the ballot secrecy game as follows:

1) BALLOT SECRECY (BS)

We say that a decentralized voting scheme is BS-secure if no probabilistic-polynomial-time adversary \( A \) that interacts with a challenger \( C \) has a non-negligible advantage in the following BS game:

- **Setup**: A setup phase of a voting system is operated, and \( C \) gives a list of eligible \( n \) voters and \( k \) candidates to \( A \).
- **Challenge**: \( A \) outputs two voters \((l_0, l_1)\) among \( n \) voters and one candidate \( j \) among \( k \) candidates. \( A \) gives \((l_0, l_1, j)\) to \( C \). Subsequently, \( C \) randomly chooses one voter among \((l_0, l_1)\) who votes for the candidate \( j \). (Naturally, such a selection indicates that the other voter should not vote for \( j \) and cast a vote for one of the other candidates except \( j \)). We refer to the voter who voted for \( j \) as voter*. 
- **Voting Round**: \( A \) can request to start the voting round. On behalf of all corrupted voters (except for \( l_0 \) and \( l_1 \)), \( A \) interacts with \( C \) to perform the voting round.
- **Guess**: \( A \) outputs a guess, \( guess \in \{l_0, l_1\} \).

We define the advantage of \( A \) in winning the abovementioned BS game as \( Adv^\text{BS}_{A} = \Pr[guess = voter^*] - 1/2 \), meaning that \( A \) must correctly guess voter* among \((l_0, l_1)\) with probability greater than 1/2.

**Definition 5.1**: A decentralized voting scheme is BS-secure if the advantage \( Adv^\text{BS}_{A} \) is negligible for all probabilistic-polynomial-time adversaries \( A \).

We now prove the ballot secrecy of our OV-Net variant using a hybrid argument in which the original security game, \( Game_0 \), is gradually changed into the final game, \( Game_F \). 

**Theorem 5.2**: Our OV-Net variant is BS-secure under the extractability of NIZK\(_1\), the zero-knowledge property of all NIZK proofs, and the DDH assumption in the random oracle model.

**Proof**: We create a series of games from the initial game, \( Game_0 \), to the final game, \( Game_F \), as follows:

- **Game_0**: \( Game_0 \) is the original BS game where the adversary \( A \) tries to break ballot secrecy in real attack scenarios.
- **Game_1**: \( Game_1 \) is the same as \( Game_0 \), except that \( C \) extracts \( x_{l,j} \) for each corrupted voter \( l \in [n] \setminus \{l_0, l_1\} \) from its corresponding public voting key using the extractor \( E \) (described in Section II-C). In this case, \( Game_1 \) is indistinguishable from \( Game_0 \) by the extractability property of NIZK\(_1\); specifically, \( C \) sequentially extracts \( x_{l,j} \) as \( A \) publishes public voting key \( X_{l,j} \) and a proof \( \pi_{l,j} \). \( C \) stores \( x_{l,j} \) to the list, \( List^C \). Consequently, all private voting keys \( x_{l,j} \) for all \( l \in [n] \setminus \{l_0, l_1\} \) are stored in \( List^C \) before beginning Round 2.
- **Game_2**: \( Game_2 \) is the same as \( Game_1 \), except that \( C \) simulates all NIZK proofs (i.e., NIZK\(_1\), NIZK\(_2\), and NIZK\(_3\)) regarding only two target voters, \( l_0 \) and \( l_1 \). In this case, owing to the zero-knowledge property of these NIZK proofs, a simulator \( S \) (described in Section II-C) exists for each NIZK proof in the random oracle model, and hence \( Game_2 \) is indistinguishable from \( Game_1 \).
- **Game_F**: \( Game_F \) is the same as \( Game_2 \), except that when computing \( U_{0,j} \) and \( U_{1,j} \), the term \( g^{x_{l,j}} \) is replaced with \( g^v \) for a randomly chosen \( v \in \mathbb{Z}_p \), which is independent of any vote value \( v_{l_0,j} \) or \( v_{l_1,j} \). In \( Game_F \), it is evident that \( A \) has no information on the vote values that the two target voters select. Therefore, to achieve this, we demonstrate the following claim.

**Claim 2**: \( Game_F \) is indistinguishable from \( Game_2 \) under the DDH problem.

**Proof**: Given an instance \((g, g^a, g^b, T)\) of the DDH problem, \( C \) sets \( X_{l_0,j} = g^{a} \) and \( X_{l_1,j} = g^{b} \) as the public voting keys for \( l_0 \) and \( l_1 \); implicitly setting \( x_{l_0,j} = a \) and \( x_{l_1,j} = b \) and generates two simulated NIZK\(_1\) proofs \((\pi_{l_0,j}, \pi_{l_1,j})\), corresponding to \( X_{l_0,j} \) and \( X_{l_1,j} \), respectively. For \( l_0 \) and \( l_1 \), the other private voting keys for other candidates are randomly generated by \( C \). Additionally, excluding \( l_0 \) and \( l_1 \), \( C \) knows all the private voting keys of other corrupted voters for \( C \) using the extractor of NIZK\(_1\). In Round 2, \( C \) needs to compute \( U_{0,j} \) and \( U_{1,j} \) using \( g^a \) and \( g^b \) as follows: We assume that \( l_0 > l_1 \) without the loss of generality.

- In case of \( U_{0,j} \): \( C \) should compute \((g^{x_{l,j}})^{y_{l_0,j}} \cdot g^{y_{l_1,j}}\) where \( y_{l_0,j} \in \{0, 1\} \) is randomly chosen by \( C \), and in performing this, we observe that

\[
\begin{align*}
(g^{x_{l,j}})^{y_{l_0,j}} & = g^{x_{l,j}(x_{l_0,j}+\cdots+x_{l_1,j}+\cdots+x_{l,n,j})} \\
& = g^{x_{l,j}x_{l_0,j}+\cdots+x_{l_0,j}x_{l_1,j}+\cdots+x_{l_0,j}x_{l,n,j}},
\end{align*}
\]

in which the term \( g^{x_{l,j}x_{l_0,j}} \) is replaced with \( T \). The other terms \( g^{x_{l,j}x_{l_0,j}} \) for all \( l \in [n] \setminus \{l_0, l_1\} \) are
easily computed using the extracted private voting keys in ListC.

- In case of $U_{l,j} \neq C$, should compute $(g^{v_{l,j}})^{v_{l,j}} g^{v_{l,j}}$, where $v_{l,j} \in \{0, 1\}$ is automatically determined by the 1’s complement of $v_{l,j}$. As described above, we observe that

$$
\begin{align*}
(g^{v_{l,j}})^{v_{l,j}} &= g^{v_{l,j} + (x_{l,j} + \ldots + x_{l-1,j} - (x_{l,j} + \ldots + x_{0,j}))} \\
&= g^{x_{l,j} x_{l,j} \ldots g^{x_{l,j} x_{l,j} - 1, j} g^{x_{l,j} x_{l,j} - 1, j} \ldots g^{x_{l,j} x_{l,j} - 1, j},
\end{align*}
$$

in which the term $g^{-x_{0,j} x_{l,j}} = g^{-a b}$ is also replaced with $T^{-1}$ and the other terms $g^{x_{l,j} x_{l,j}}$ for all $l \in [n] \setminus \{l_{0}, l_{1}\}$ are easily computed as above.

One important thing to note is that $\prod_{l=1}^{n} g^{x_{l,j} x_{l,j}} = 1$ still holds even when the term $g^{x_{0,j} x_{l,j}} = g^{-a b}$ is replaced with $T$ because of the cancellation, such as $T \cdot T^{-1} = 1$ during the multiplications. Additionally, $C$ computes $V_{l_{0,j}}$ and $V_{l_{1,j}}$, because $C$ knows $x_{l_{0,j}}$ and $x_{l_{1,j}}$ for all $i \in [K] \setminus \{l_{0}, l_{1}\}$. Moreover, $C$ can generate all simulated NIZK2 and NIZK3 proofs with respect to $(U_{l_{0,j}}, U_{l_{1,j}})$ and $(V_{l_{0,j}}, V_{l_{1,j}})$, respectively, although $C$ does not know all the relevant witnesses.

In the Guess phase, $A$ outputs its guess, $\text{guess} \in \{l_{0}, l_{1}\}$. If $\text{guess} = \text{voter}^*$, $C$ outputs 1, indicating that $T = g^{a b}$. Otherwise, $C$ outputs 0, indicating that $T$ is random.

When $T = g^{a b}$, $A$ is in Game2 where $U_{l_{0,j}}$ and $U_{l_{1,j}}$ values are correctly generated, while hiding the votes $v_{l_{0,j}}$ and $v_{l_{1,j}}$, respectively. Therefore, the ability of $A$ to break the OV-Net variant is transferred to that of $C$ to break the DDH problem. Conversely, when $T = g^{e}$ for a random exponent $e \in \mathbb{Z}_{p}$, $A$ is in Game3 where $U_{l_{0,j}}$ and $U_{l_{1,j}}$ values are falsely generated, regardless of the votes $v_{l_{0,j}}$ and $v_{l_{1,j}}$. In this case, it is difficult for $A$ to obtain any information about the votes; thus, the advantage of $A$ in Game3 is negligible.

This completes the proof of Theorem 5.2.

### B. DISPUTE FREENESS

As previously mentioned, dispute-freeness demonstrates that it is difficult for an adversary $A$ to collude for breaking the one-man-one-vote rule without being detected. We consider the full collusion by which $A$ can corrupt all voters. Intuitively, our OV-Net variant forces $A$ to follow the one-man-one-vote rule by the soundness property of two NIZK2 and NIZK3 and through the publicly verifiable equation check of $\prod_{l=1}^{n} V_{l_{i,l}} = g$ for each voter $l \in [n]$ because any attempt to cast more than one ballot is easily detected by failing to either verify the NIZKs or pass the equation check.

1) DISPUTE-FREENESS (DF)

We say that a decentralized voting scheme is dispute-free if no probabilistic-polynomial-time adversary $A$ that interacts with a challenger $C$ has a non-negligible advantage in the following dispute-freeness game:

- **Setup**: A setup phase of a voting system is operated, and $C$ gives $A$ a list of eligible $n$ voters and $k$ candidates.

- **Voting Round**: $A$ can request to start the voting round. On the behalf of all corrupted voters, $A$ interacts with $C$ to conduct the voting round.

We say that $A$ wins in the abovementioned DF game if at least one of the (corrupted) voters successfully casts more than one ballot to any candidate without being detected. The advantage of $A$ winning the abovementioned game is then defined as $\text{Adv}_{A}^{DF} = \text{Pr}[\text{A wins}]$.

**Definition 5.1**: A decentralized voting scheme is dispute-free if the advantage $\text{Adv}_{A}^{DF}$ is negligible for all probabilistic-polynomial-time adversaries $A$.

For the proof of the ballot secrecy, we prove the dispute-freeness of our OV-Net variant using a hybrid argument in which the original security game Game0 is gradually changed into the final game GameF. In GameF, $A$ cannot break the one-man-one-vote rule.

**Theorem 5.2**: Our OV-Net variant is dispute-free under the extractability of NIZK1 and the soundness properties of NIZK2 and NIZK3 in the random oracle model.

**Proof**: We create a series of games as follows:

- **Game0**: Game0 is the original DF game where adversary $A$ tries to break the dispute-freeness in real attack scenarios.

- **Game1**: Game1 is the same as Game0, except that $C$ extracts all private voting keys $x_{l_{i,l}}$ for all the corrupted voters $l \in [n]$ and all the candidates $i \in [k]$ using the extractor $E$ of NIZK1. In this case, Game1 is indistinguishable from Game0 owing to the extractability property of NIZK1. All private voting keys are stored in ListC before starting Round 2.

- **Game2**: Game2 is the same as Game1, except that $A$ outputs any false statement and its relevant NIZK2 proof such that the verification succeeds. Recall that a true statement with respect to NIZK2 is in the form of $(X_{l_{i,l}} = g^{x_{l_{i,l}}}, U_{l_{i,l}} = (Y_{l_{i,l}}) g^{x_{l_{i,l}}}, V_{l_{i,l}} = (Z_{l_{i,l}}) g^{x_{l_{i,l}}})$ for all $l \in [n]$ and $i \in [k]$. Because $C$ knows all the private voting keys of voters, $C$ can verify if a given statement along with a corresponding NIZK2 is in the correct form by reconstructing the statement. If a false statement and its NIZK2 proof exist, $C$ outputs them as a solution to break the soundness of NIZK2. Therefore, Game2 is indistinguishable from Game1.

- **Game3**: Game3 is the same as Game2, except that $A$ outputs any false statement and its relevant NIZK3 proof such that verification succeeds. Similar to the above-mentioned analysis, $C$ can check whether a given statement along with a corresponding NIZK3 is in the correct form of $(U_{l_{i,l}} = (Y_{l_{i,l}}) g^{x_{l_{i,l}}}, V_{l_{i,l}} = (Z_{l_{i,l}}) g^{x_{l_{i,l}}})$. If a false statement and its NIZK3 proof exist, $C$ outputs them as a solution to break the soundness of NIZK3. Therefore, Game3 is indistinguishable from Game2.

- **GameF**: GameF is the same as Game3, except that the sum of the votes cast by a voter for all $k$ candidates is not equal to one. Given the correct statement of NIZK3 with respect to the values $\{V_{l_{i,l}}\}_{i=1}^{k}$,
Table 3: Theoretical performance comparison.

|                  | OV-Net [8] | Ours          |
|------------------|------------|---------------|
| Voting           | \((4k + 2)\) E | 14k E         |
| Verifying        | \((4k + 2)(n - 1)\) E | \((16k + 4)(n - 1)\) E |
| Tallying         | \(n^2(2k-1)^m\) M | \((n + 1)k\) M |
| Transmission size| \((2k + 4)\) | \([|G| + |H|]\) |

\(n\): number of voters, \(k\): number of candidates, \(m\): least integer such that \(2^m > n\), \(E\): exponentiation, \(M\): multiplication, \(|G|\): the size of group element in \(G\), \(|H|\): the size of the hash output.

The product of the \(\prod_{i=1}^{k} V_{i,l}\) becomes \(g^{\sum_{i=1}^{n} v_{i,l}}\) for each voter \(l \in [n]\) by the same technique of OV-Net. This is due to the fact that with the publicly-computed elements \(\{Z_{i,l}\}_{i=1}^{k}\), the equality of discrete logarithms of each pair \((g^{v_{l,i}}, (Z_{i,l})^{v_{l,i}})\) is proved by the soundness of NIzk2 and NIzk3. Unless the soundness is not broken by \(A\), we have the result \(\prod_{i=1}^{k} (Z_{i,l})^{v_{l,i}} = 1\); therefore, \(\prod_{i=1}^{k} V_{i,l} = \prod_{i=1}^{k} (Z_{i,l})^{v_{l,i}} g^{v_{l,i}} = g^{\sum_{i=1}^{n} v_{i,l}}\). Because the equation \(\prod_{i=1}^{k} V_{i,l} = g\) holds for each voter \(l \in [n]\), \(\sum_{i=1}^{k} v_{i,l} = 1\) with probability 1 for a polynomial number \(n\) of voters much smaller than the group order \(p\). Thus, GameF is identical to Game3.

Indeed, the equation \(\prod_{i=1}^{k} V_{i,l} = g\) can be publicly verified for each voter such that when inequality happens, anyone can immediately see, without dispute, which voter broke the one-man-one-vote rule. In GameF where equality holds for all voters, \(A\) has no choice but to comply with the rule on behalf of all voters. Therefore, this concludes the proof of Theorem 5.3.

VI. PERFORMANCE COMPARISON

We present a performance comparison between the traditional OV-Net [8] and ours for multiple candidates, particularly regarding the overall computational cost for voting and tallying. Although there have been several variants [10], [11], [12], [13], [14], [23] of the OV-Net [8], we select only [8] for comparison because all these variants are based on the same tallying technique as employed in [8] using the super increasing number (described as the third suggestion in III-B); Additionally, the overall computational cost is overwhelmingly dominated by tallying. We provide the performance comparison, depending on the number of voters and candidates.

A. THEORETICAL COMPARISON

1) VOTING COST

A voting cost comprises the cost for generating and verifying NIZK proofs. For \(n\) voters and \(k\) candidates, each voter in the OV-Net [8] must generate a NIZK proof for publishing its public voting key, compute a 1-out-of-\(k\) NIZK proof for hiding its vote, and verify such NIZK proofs generated from the other \((n - 1)\) voters. Conversely, each voter in ours is required to generate a NIZK1 proof for publishing its public voting key, a 1-out-of-2 NIZK2 proof for hiding its vote, and a NIZK3 proof for proving the equality of discrete logarithms, all of which are generated per each candidate. Thus, each voter in ours must verify all \((n - 1)\) \{NIZK1, NIZK2, NIZK3\} proofs generated from other voters. Without considering marginal operations such as group multiplication and hash evaluation, the voting costs of the traditional OV-Net and ours are compared in Table 3. For the same number of \(n\) voters and \(k\) candidates, the voting cost of ours is more expensive than that of the traditional OV-Net. For instance, if \(n = 1024\) and \(k = 5\), our voting requires approximately \(3.8(= 86, 002/22, 528)\) times more exponentiations than the traditional OV-Net.

Table 3 shows that the transmission size depends only on the number of candidates. The transmission size presented in the table comprises all the NIZK statements and proofs that each voter should broadcast to the voting network. When assuming that \(|G| = |H|\), setting \(k = 5\) shows that ours must transmit values approximately \(4.3(= 13k/(2k + 5))\) times larger than the values transmitted by the traditional OV-Net.

2) TALLYING COST

Compared to the traditional OV-Net, both the voting cost and the transmission size make our OV-Net variant more inefficient. However, the tallying cost dominates the overall running time that is sufficient to offset the inefficiency caused by the two factors. The tallying cost is measured by the number of group multiplications required to solve a discrete logarithm problem using an exhaustive search. Table 3 shows that the OV-Net requires \(n^2(k-1)^m\) multiplications, where \(m\) is determined by \(2^m > n\), whereas ours requires \((n + 1)k\) multiplications. For simplicity, when we set \(2^m \approx n\) as \(m \approx \log n\), then the number of multiplications in the OV-Net becomes approximately \(n^k\). In the abovementioned case, when \(n = 1024\) and \(k = 5\), the tallying in ours simply needs to perform \(5,125\) multiplications, but the tallying in the traditional OV-Net should perform approximately \(2^{50}\) multiplications, which are approximately \(2^{57}(= 2^{50}/5, 125)\) times more multiplications than ours. Additionally, for comparison, when we roughly set \(2^m\) multiplications to be equivalent to one exponentiation, \(2^{50}\) multiplications implies \(2^{41}\) exponentiations, which would be a burden for tallying. Consequently, when considering both the voting and the tallying costs, we can expect that ours become more highly efficient than the traditional OV-Net as the numbers \(n\) and \(k\) increase.

B. EXPERIMENTAL COMPARISON

To measure the total times spent by voting and tallying, we first choose a group \(G\) and a generator \(g\) based on the elliptic curve SECP256K1 [24], targeting the current 128-bit security level. Then, we perform experiments according to the number of \(n\) voters and \(k\) candidates on a Desktop running windows 10 equipped with an Intel(R) Core(TM) i7-8700K CPU 3.70-GHz and 16-GB RAM. Specifically, we chose SECP256K1 for our experiment among various elliptic curves considering our future work related to Ethereum that uses SECP256K1 as the core of its trusted infrastructure (cf. VII-B).
1) SCALABILITY ACCORDING TO THE NUMBER OF CANDIDATES
To determine the time consumed for voting and tallying as the number $k$ of candidates increases, we fix the total number $n$ of voters as 40, as in previous works [11], [13], [14], and gradually increase the number $k$ of candidates one by one. Next, we assume that each candidate receives an approximately equal number of votes for the average tallying time per voter. Figures 6 and 7 present the total time required for each voter to complete the voting and tallying using the traditional OV-Net and ours, respectively. As expected, when $n (= 40)$ is fixed, the time for ours increases linearly with $k$, whereas the time for the traditional OV-Net increases exponentially with $k$. Therefore, a voter of ours takes approximately 31 ms when the number of candidates is 15, while a voter using the traditional OV-Net takes approximately 43 ms when the number of candidates is 3. Figure 6 and 7 are plotted in Figure 8 on a logarithmic scale with base 2, and Figure 8 clearly shows how the time difference between the two voting systems changes with increasing $k$. Specifically, in the case of the OV-Net, we experimented only when there were five or fewer candidates because the performance time had already reached a sufficiently high level (e.g., 100,000 ms) when there were five candidates.

2) SCALABILITY ACCORDING TO THE NUMBER OF VOTERS
To examine scalability according to the number $n$ of voters, we perform the experiment under the settings where the number $k$ of candidates is fixed at 3 and the number $n$ increases by 50 until it becomes 500. As in the above experiment, we assume that each candidate receives approximately equal number of votes. Therefore, for fixed $k = 3$, we can expect, from Table 3, that the number of multiplications required for tallying is $O(n)$ in ours; however, $O(n^3)$ in the OV-Net (when setting $m \approx \log n$). Figures 9 and 10 show the time required for each voter to perform voting and tallying using the traditional OV-Net and ours, respectively. Figure 9 and 10 are plotted in Figure 11 on a logarithmic scale with base 2, and Figure 11 clearly shows the time difference between the traditional OV-Net and ours, which increases drastically with increasing $n$. Notably, the rate of increase in the traditional OV-Net becomes steeper than that in ours. Furthermore, increasing the number $k$ of candidates is fatal in the traditional OV-Net, in which case the tallying of the traditional OV-Net must determine a much larger discrete logarithm $\gamma$ such that $g^\gamma = g^{2^{c_1} + 2^{c_2} + \ldots + 2^{k-1}c_k} \bmod n$ (as described in Section III-B3), where increasing both $k$ and $n < 2^m$ causes a significant expansion of possible exponential values.

VII. DISCUSSION
Despite solving the scalability, our voting system has limitations in terms of fairness, robustness, and coercion-resistance similar to the previous OV-Net. In this section, we introduce several studies conducted to overcome these limitations, all of which can also be applied to our voting system.
A. FAIRNESS

The notion of fairness means that a tallying result must be kept secret before every voter casts his/her vote. Similar to the traditional OV-Net, ours does not provide fairness to the last voter. This is because for \( n \) voters, the last \( n \)-th voter can know the tallying result in advance by simulating his/her own vote \( g^{\text{vote}_i} \) for all candidates \( i \in [k] \) when all the \((n-1)\) ballots ahead of the last voter are published. Failure to provide the last voter with the fairness property can lead to adaptive and abortive issues [11]. The adaptive issue is the issue where the previous knowledge of the last voter can influence his/her vote in his/her favor, and the abortive issue is the one where the last voter gives up his/her vote so that no one can proceed with the final tally.

To solve the adaptive issue, Kiayias and Yung [6] and Groth [7] proposed a voting system where an election commission as the last voter casts the ballot that will be excluded from the final tally. However, their method reintroduces a trusted third party, which is assumed to obey the voting rule and not collude with malicious voters. To solve the adaptive issue without a trusted third party, McCorry et al. [11] optionally add a commitment procedure to Round 2 in the OV-Net. Specifically, each voter publishes his/her commitment \( H(g^{\text{vote}_i}) \), where \( H \) is a secure hash function, before casting his/her votes; as a result, no one can change his vote, even when the last voter can precompute the tally.

B. ROBUSTNESS

The notion of robustness means that a voting system is resistant to malicious voters blocking the completion of the voting process up to a self-tallying. Including the aforementioned abortive issue, a representative action of breaking the robustness is that a malicious voter intentionally gives up a vote in the middle of the voting phase. Essentially, the adversarial action cannot be easily prevented because of the design principle of the OV-Net, which requires all voters to vote cooperatively to count the votes.

McCorry et al. [11] proposed the blockchain-based OV-Net using a smart contract over the Ethereum network to mitigate the robustness problem. In their construction, all voters must deposit money into a smart contract to register for an election in Round 1, and only a voter who casts a ballot in Round 2 can get a refund. Here, the smart contract acts as a means of enforcing malicious adversaries to execute a voting protocol normally by not returning the money to them. However, this method is an indirect solution that relies on monetary sanctions and does not prevent an attacker who is willing to take the monetary loss for breaking the robustness.

C. COERCION-RESISTANCE

Coercion-resistance means that a voter can vote for any candidate he/she chooses even when a coancer exists who forces the voter to cast a vote for a specific (unwanted) candidate. However, in a decentralized environment, ensuring such coercion-resistance in a cryptographic manner is very
difficult without a trusted third party. Therefore, our system is considered suitable for low-coercion elections where no coercion is assumed, as in several other decentralized voting systems [6], [7], [8], [11].

VIII. CONCLUSION

Herein, we presented a variant of the OV-Net that can be used much more efficiently in situations where more than three candidates exist, whether the number of voters is small or large; this has been demonstrated through our experiments. Further, we proved that our system fulfills ballot secrecy under partial collusion without assuming honest-but-curious voters and dispute-freeness under full collusion. Therefore, the proposed system is considerably more efficient than the traditional OV-Net when three or more candidates exist and as secure as the traditional OV-Net.

However, our system also has the limitations of fairness, robustness, and coercion-resistance as observed in the traditional OV-Net, although the study results in addressing the limitations of fairness and robustness (the limitation of coercion-resistance requires further research such as how to define coercion-resistance from a cryptographic perspective in a decentralized environment where no external help from a trusted third party can be obtained) in the traditional OV-Net can also be applied to our system. However, as described in Section VII, the results also have limitations. Therefore, a future study to completely guarantee fairness and robustness in our system is pertinent.

REFERENCES

[1] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” Comm. ACM, vol. 24, no. 2, pp. 84–90, 1981.

[2] Z. Xia, S. Schneider, J. Heather, P. Y. Ryan, D. Lundin, R. Peel, and P. Howard, “Prêta voter: All-in-one,” in Proc. IAViSS Workshop Trustworthy Elections (WOTE), 2007, pp. 47–56.

[3] P. Y. Ryan and V. Teague, “Pretty good democracy,” in Proc. Int. Workshop Secur. Protocols. Berlin, Germany: Springer, 2009, pp. 111–130.

[4] A. Juel, D. Catalano, and M. Jakobsson, “Coercion-resistant electronic elections,” in Towards Trustworthy Elections. Berlin, Germany: Springer, 2010, pp. 37–63.

[5] B. Adida, “Helios: Web-based open-audit voting,” in Proc. USENIX Secur. Symp., vol. 17, 2008, pp. 335–348.

[6] A. Kiayias and M. Yung, “Self-tallying elections and perfect ballot secrecy,” in Proc. Int. Workshop Public Key Cryptogr. Berlin, Germany: Springer, 2002, pp. 141–158.

[7] J. Groth, “Efficient maximal privacy in boardroom voting and anonymous broadcast,” in Proc. Int. Conf. Financial Cryptogr. Berlin, Germany: Springer, 2004, pp. 90–104.

[8] F. Hao, P. Y. Ryan, and P. Ziemlinski, “Anonymous voting by two-round public discussion,” IET Inf. Secur., vol. 4, no. 2, pp. 62–67, 2010.

[9] F. Hao and P. Ziemlinski, “A 2-round anonymous veto protocol,” in Proc. Int. Workshop Secur. Protocols. Berlin, Germany: Springer, 2006, pp. 202–211.

[10] D. Khader, B. Smyth, P. Ryan, and F. Hao, “A fair and robust voting system by broadcast,” Lect. Notes Comput. Sci., vol. 205, pp. 285–299, Jul. 2012.

[11] P. McCorry, S. F. Shahandashti, and F. Hao, “A smart contract for boardroom voting with maximum voter privacy,” in Proc. Int. Conf. Financial Cryptogr. Data Secur. Cham, Switzerland: Springer, 2017, pp. 357–375.

[12] Y. Lin and P. Zhang, “Blockchain-based complete self-tallying E-voting protocol,” in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC), Nov. 2019, pp. 47–52.

[13] M. Seifelnasr, H. S. Galal, and A. M. Youssef, “Scalable open-vote network on ethereum,” in Proc. Int. Conf. Financial Cryptogr. Data Secur. Cham, Switzerland: Springer, 2020, pp. 436–450.

[14] M. ElSheikh and A. M. Youssef, “Dispute-free scalable open vote network using zk-SNARKs,” 2022, arXiv:2203.03363.

[15] R. Cramer, I. Damgård, and B. Schoenmakers, “Proofs of partial knowledge and simplified design of witness hiding protocols,” in Proc. Annu. Int. Cryptol. Conf. Berlin, Germany: Springer, 1994, pp. 174–187.

[16] D. Boneh, “The decision Diffie–Hellman problem,” in Proc. Int. Algorithmic Number Theory Symp. Berlin, Germany: Springer, 1998, pp. 48–63.

[17] M. Blum, P. Feldman, and S. Micali, “Non-interactive zero-knowledge and its applications,” in Proc. 20th Annu. ACM Symp. Theory Comput. (STOC). New York, NY, USA: Association for Computing Machinery, 1988, pp. 103–112.

[18] U. Feige, D. Lapidot, and A. Shamir, “Multiple non-interactive zero knowledge proofs based on a single random string,” in Proc. 31st Annu. Symp. Found. Comput. Sci., 1990, pp. 308–317.

[19] D. Pointcheval and J. Stern, “Security proofs for signature schemes,” in Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 1996, pp. 387–398.

[20] E.-J. Goh, S. Jarecki, J. Katz, and N. Wang, “Efficient signature schemes with tight reductions to the Diffie–Hellman problem,” J. Cryptol., vol. 20, no. 4, pp. 493–514, Oct. 2007.

[21] R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” Eur. Trans. Telecommun., vol. 8, no. 5, pp. 481–490, Sep. 1997.

[22] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, “Practical multi-candidate election systems,” in Proc. 20th Annu. ACM Symp. Princ. Distrib. Comput. (PODC), 2001, pp. 274–283.

[23] G. Bana, M. Biroli, M. Dervishi, F. E. Orche, R. Géraud-Stewart, D. Naccache, P. B. Ronne, P. Y. A. Ryan, and H. Waalburger, “Time, privacy, robustness, accuracy: Trade offs for the open vote network protocol,” IACR Cryptol. ePrint Arch., p. 1065, Sep. 2021. [Online]. Available: https://eprint.iacr.org/2021/1065

[24] S. Blake-Wilson and M. Qu, “Standards for efficient cryptography (SEC) 2: Recommended elliptic curve domain parameters,” Certicom Res., Mississauga, ON, Canada, Oct. 1999.

SANGSOO SEOL received the B.S. degree from the Department of Information Security, Sejong University, Seoul, South Korea, in 2021. He is currently pursuing the M.S. degree with the Graduate School of Information Security, Korea University. His research interests include electronic voting systems and cryptographic protocols.

HYOSEUNG KIM received the B.S. degree from the Department of Mathematics, Korea University, Seoul, South Korea, in 2010, and the Ph.D. degree from the Graduate School of Information Security, Korea University, in 2021. Since 2021, he has received funding from Brain Korea 21(BK21, 4th step) as a Postdoctoral Researcher. His research interests include anonymous signatures and zero-knowledge proofs.

JONG HWAN PARK received the B.S. degree from the Department of Mathematics, Korea University, Seoul, South Korea, in 1999, and the M.S. and Ph.D. degrees from the Graduate School of Information Security, Korea University, in 2004 and 2008, respectively. From 2009 to 2011, he was a Research Professor with Kyung Hee University, and from 2011 to 2013, he was a Research Professor with Korea University. Since 2013, he has been an Assistant Professor with the Department of Computer Science, Sungkyunkwan University, Seoul. His research interests include functional encryption, broadcast encryption, authenticated encryption, and various cryptographic protocols.

* * *