Hindawi
Security and Communication Networks
Volume 2022, Article ID 4436824, 15 pages
https://doi.org/10.1155/2022/4436824

Research Article

A Practical Anonymous Voting Scheme Based on Blockchain for Internet of Energy

Houpeng Hu,[1] Jiaxiang Ou,[1] Bin Qian,[2] Yi Luo ,[2] Peilin He,[1] Mi Zhou,[2] and Zerui Chen[1]

[1]*Guizhou Power Grid Co. Ltd, Guiyang, China*
[2]*Institute of Metrology Technology Electric Power Research Institute CSG, Guangdong Provincial Key Laboratory of Intelligent Measurement and Advanced, Metering of Power Grid, Guangzhou, China*

Correspondence should be addressed to Yi Luo; luoyi_csg@outlook.com

Received 26 April 2022; Revised 15 June 2022; Accepted 5 July 2022; Published 21 August 2022

Academic Editor: Zheng Yang

E-voting allows us to build a democratic business in most Internet of things (IoT) systems. For example, we may vote to choose a proper energy broker in a smart grid system. In this study, we focus on e-voting services in an Internet of energy (IoE) system, which is a new-style smart grid. A practical e-voting in IoE may focus on the properties of fairness, decentralization, eligibility, anonymity, compatibility, verifiability, and coercion resistance. It is difficult to fulfil all these properties simultaneously. Traditional voting schemes often use a public bulletin board or administrator in the voting process, which makes them become centralized. Services that offer e-voting via blockchain can make the voting schemes decentralized. However, many of them ignore the complexity of organizing the data of the transactions, which should be confirmed by the miners. Moreover, to the best of the authors' knowledge, no works have tested the performance in the blockchain while considering practical use cases and constraints. Concerning all the challenges, we propose a practical anonymous voting scheme for IoE called IoEPAV. The proposed scheme fulfils all the mentioned design goals simultaneously. We tested IoEPAV both in different test networks of the Ethereum blockchain to give an overall evaluation. The practical evaluation can show that the proposed scheme is easy to be integrated into a real system like IoE. We also gave a comparison analysis with the state-of-the-art blockchain-based e-voting. All the results show that IoEPAV is decentralized, verifiable, anonymous, and highly efficient.

1. Introduction

The rapid development of the Internet of things (IoT)[1] promotes the development of various new intelligent network systems, i.e., smart power grid system [2]. Smart grid systems have led to a modern power network called the Internet of energy (IoE), which has drawn great interest from many countries [3]. As a kind of IoT, IoE is reshaping the energy industry into a smart industry with features of data-driven decision-making. However, the intrinsic features of IoE raise a number of challenges, such as autonomy, privacy, and decentralization [4–6]. One of the most common activities/applications in IoE is voting to make a decision.

In this study, we focus on the problem of designing a voting scheme fit for an IoE system. For instance, we may vote to choose a proper energy broker in IoE system. Recently, e-voting has become attractive [7] for its convenience in building a democratic activity/business. Voting schemes in an online way called e-voting have been studied by both academic world and industry world [7, 8]. Here, we formally state the design goals, which a deemed secure e-voting scheme must hold [9]. Moreover, we extend the design goals, which particularly are required in IoE [4]. All of them are as follows:

(i) Fairness. Fair voting should assure that no one can obtain the ballot results of others before he/she has submitted his/her ballot. It means the choice of a voter cannot be influenced by those who have voted ahead.

(ii) Decentralization. Any kind of trusted third party (TTP) such as election administrators and (independent) observers should be eliminated from the voting scheme.

(iii) Eligibility. The right of a voter should be checked before he/she begins to vote. To address this, most voting schemes will verify the identities of the voters at the beginning. Moreover, every voter can cast their vote only once.

(iv) Anonymity. The privacy of voters should be protected to make sure that no one can know the owner of a ballot from the voting result at the end.

(v) Compatibility. The voting scheme should be as simple as possible to be integrated into an IoE system.

(vi) Verifiability. Contrasted with the "Anonymity" property, verifiability guarantees that all the stages of the voting can be audited by the voters. For instance, a voter should be able to check whether his/her vote has been tallied or not. Moreover, the validity of each vote should be able to be verified by anyone. It seems to be a contradiction of the "Anonymity" property, and it is difficult for a voting scheme to acquire the two properties at the same time. We will show how to address this in our scheme later.

(vii) Coercion Resistance. To avoid anyone trying to coerce the voters to vote by following their instructions, a voting scheme should be coercion-resistant.

Fairness and eligibility are essential properties, which any voting scheme should fulfil. Besides, it is important to handle voting without any kind of trusted third party (TTP) in IoE due to its open and distributed features. This is a challenge for traditional e-voting schemes [10–12]. Most of them assumed that there are administrators or authorities implemented by a Web server to provide a consistent view of the results. As a result, a trusted third party is involved. Unfortunately, with a trusted third party, the protocols will be subjected to the single point of failure and are not available for a trustless environment.

Fortunately, blockchain technology offers a novel way to address the challenges of IoE [13]. There are already voting schemes built based on a blockchain network [14–16]. However, new challenges are raised to design a decentralized voting protocol via blockchain in IoE [13, 17, 18]. Firstly, it is difficult to provide verifiability along with anonymity, which seems two contradictory design goals for blockchain. Secondly, the voting scheme as a basic service in IoE should be compatible with the system. Some of the proposals assume that a voter can organize the data structure of a transaction unboundedly and it seems impossible for the existed blockchain networks. We say they are not practical. Thirdly, most works use theoretical analysis without real-world generalizable experiments, and to the best of our knowledge, no works have tested the performance while considering practical use cases and constraints.

Briefly, it is hard to find a solution that fulfils all the design goals mentioned above. To address this, our contributions are summarized as follows:

(i) We propose a blockchain-based decentralized anonymous voting scheme for the Internet of energy. To the best of our knowledge, our voting scheme called IoEPAV is the first work to take the key features of the IoE into account. Theoretical analysis is given to show that our proposed scheme fulfils the seven formally stated design goals and approaches to resist all the attacks in the threat models.

(ii) To make the proposed scheme practical, we use smart contracts to automate the voting process of the Internet of energy. With smart contracts, the voting scheme can be easy to integrate into the IoE system. A voter can follow the voting protocol by invoking the interfaces of the smart contracts. Any blockchain system including Ethereum 2.0, which supports smart contracts, is feasible for the proposed scheme, and we do not need to construct a whole new blockchain platform.

(iii) Compared with Yang's state-of-the-art blockchain-based scheme, our scheme enjoys both decentralization and fewer cryptographic operations; thereafter, we conduct experiments both in the development network and two live testnet of the Ethereum blockchain and the experiment shows that we have implemented a simple, effective, accurate, and low-cost decentralized trusted anonymous voting scheme.

The rest of this study is arranged as follows. In Section 2, we give the related work of the e-voting service. In Section 3, we introduce the necessary preliminary knowledge. In Section 4, we give our system model and security analysis. In Section 6, we provide an evaluation of the development network and testnet of the Ethereum blockchain. Finally, in Section 7 we draw a brief conclusion.

2. Related Work

To address the problem of large-scale elections, Fujioka et al. [12] presented a classic voting protocol in 1992. Their voting scheme is thought to be practical and solves the privacy and fairness problems. Thereafter, Ohkubo et al. [19] tried to decrease the voting round complexity to get a more convenient voting scheme for the voters than that presented by Fujioka. They introduced a kind of distributed talliers in their scheme. As an extension of the voting scheme proposed by Fujioka, a new coercion-resistant voting scheme offered by [20] in 2017 is provided and is an efficient scheme.

Since a kind of trusted third party (TTP) such as election administrators and (independent) observers was introduced in these practical voting schemes, solutions using blockchain technology have become a refreshed framework for voters to address the issues of fraud and corruption. In 2018, Srivastava et al. [21] proposed a voting model via blockchain to alleviate known problems in voting systems. Follow My Vote [14] provides a secure online voting platform based on blockchain. Follow My Vote has the capacity to audit the ballot box and watch the real-time voting progress. Another

organization Agora [15] proposed a digital voting system using blockchain, where votes will be recorded to various layers, assuring that the voting result could not have been tampered with. Braghin et al. [22] studied various consensus algorithms and cryptographic primitives such as homomorphic encryption and one-time ring signature, which solved the cryptographic problem of security conflicts, thus improving the security of voting system and making voting system more secure in a wider range.

To ensure security, privacy, and public verifiability of the whole progress, [16, 23] presented a new voting protocol that does not rely on any TTP. In [16], the authors employ a novel encryption mechanism to encrypt each vote. Proofs are generated for each encrypted vote as well. All the proofs will be stored in a blockchain, and everyone can check the validity of these proofs. They provided a performance and security analysis, which is claimed to show that the voting protocol is feasible for real-life elections. However, the implementation does not include the part interacting with the blockchain, and we cannot see the results in a real system. Table 1 gives an overall comparison of similar systems.

3. Preliminaries

3.1. Commitments and Voting. Commitment also called cryptographic commitment [24] is an important cryptographic primitive that has many applications. Here is an example to show the relationship between commitments and votes. Think about a situation that Voter 1 and Voter 2 decide to participate in a voting behavior along with others. In this scenario, the voting institution responsible for counting the votes uses a sealed vote, which generally works as follows: each voter submits a secret sealed vote for the candidate. Once all the votes have been cast, they can be counted. The voting mechanism has good game-theoretic properties, provided that voters do not collude and do not know each other's votes until the voting close. Therefore, a sealed vote is required.

Next, we consider how to do sealed voting when some voters are of outfield and communicate with the voting institution throughout the Internet. Here, a cryptographic commitment helps. To make it simple, we assume that there are two parties in a commitment scheme Commit. Let Alice be one party, who can firstly publish a string c as a commitment for a message $m \in M$. Then, with the property of cryptographic commitment, Alice can make other party, i.e., Bob, believe that the committed message was m, by opening the commitment. Generally speaking, a cryptographic commitment Commit consists of algorithms (C, V) that

(i) On input $m \in M$, the message to be committed, Algorithm $C(m)$ outputs two strings (c, o), and we call c the commitment string and o the opening string.

(ii) On input $m \in M$ and (c, o), Algorithm V outputs accept or reject indicating whether the committed message was m.

Alice firstly inputs a message $m \in m$ and calls Algorithm C to calculate (c, o). She sends the commitment string c to Bob and keeps the opening string o secret. Later, when Alice wants to open the commitment, she sends Bob m and o. Finally, Bob can verify whether the committed message was m by running Algorithm V.

A secure cryptographic commitment scheme Commit is required to satisfy the following two properties:

(i) Binding. Binding requires that a commitment does not disclose any additional information about the message. In particular, assume the adversary $\mathscr{A}$ outputs a 5-tuple (c, m_1, o_1, m_2, o_2), and we require the advantage that $(\text{BINDadv}[\mathscr{A}, C]: = \Pr[m_1 \neq m_2)$ and $V(m_1, c, o_1) = V(m_2, c, o_2) = \text{accept}]$ is negligible.

(ii) Hiding. Hiding requires that different messages do not produce the same commitment.

We use semantic security definition to formalize this. In particular, two games are performed between an adversary $\mathscr{A}$ and a challenger, denoted as Game 0 and Game 1. Let $b = 0, 1$, in the Game b, and the adversary $\mathscr{A}$ first outputs $m_0, m_1 \in M$, inputs a message $m_b \in M$, calls Algorithm C to calculate (c, o), and passes c to $\mathscr{A}$. Finally, $\mathscr{A}$ outputs a guess $\widehat{b} \in 0, 1$. For $b = 0, 1$, W_b is defined as the case, where $\mathscr{A}$ outputs 1 in Game b. We require that the advantage that

$$\text{BINDadv}[\mathscr{A}, C]: = \left|\Pr[W_0] - \Pr[W_1]\right|, \tag{1}$$

is negligible.

3.2. Blind Signature. According to the description in [25, 26], a cryptography blind signature is a kind of digital signature where the original message should be blinded (disguised) before being signed. Then, the signer will sign on the blinded message in a way like a conventional digital signature and output a blind signature. Then, the requester can generate a corresponding signature for the original message. In the end, the signature can be verified by everyone in a way like a conventional digital signature. The technique is usually used to provide a kind of privacy protection when the message requester and signer are not the same. For example, blind signatures can be used in an election system.

In a general signature scheme illustrated in Figure 1, the signer produces a digital signature on known message content. Compared with the general signature scheme, the process of blind signature [27] is as illustrated in Figure 2. The requester performs a blinding shift on the message before sending it to the signer. The signer who then signs on the blinded message will generate a blind signature and send it to the requester.

A general signature scheme is shown in Figure 1, in which a signer can generate a digital signature on a known message. Unlike a signature scheme, the process of blind signature is shown in Figure 2, in which the requester first blinds the message before it is sent to the signer, and then, the signer signs the blind message and sends a blind signature to the requester. With the blind signature, the requester can generate an unblinded signature for the original message.

Table 1: Property comparison.

Protocol	Fairness	Decentralization	Eligibility	Anonymity	Compatibility	Verifiability	Coercion resistance
Fujioka et al. [12]	✓		✓	✓		✓	
Ohkubo et al. [19]	✓		✓	✓		✓	
Grontas et al. [20]	✓		✓	✓		✓	✓
Follow My Vote [14]	✓	✓	✓			✓	
Yang et al. [16]	✓	✓	✓	✓		✓	✓
IoEPAV	✓	✓	✓	✓	✓	✓	✓

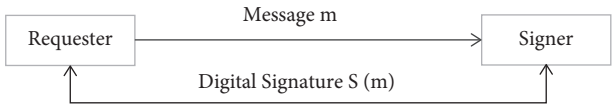

Figure 1: A general signature scheme.

In our voting scheme, we use the blind signature scheme of elliptic curve cryptography (ECC) and adopt the secp256k1 [28] elliptic curve. Elliptic curve domain parameters over $\mathbb{F}_p$ are a sextuple: $T(p, a, b, G, n, h)$ consisting of an integer p specifying the finite field $\mathbb{F}_p$ and two elements $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ defined by the following equation:

$$E: y^2 \equiv x^3 + ax + b \pmod{p}. \tag{2}$$

G is a base point on $E(\mathbb{F}_p)$, n is the order of G, h is the cofactor where $h = \#E(\mathbb{F}_p)/n$, and $\mathbb{Z}_n$ represents the integer not more than n. Suppose that (d, P) is an asymmetric key pair of the signer, the message is m, and all else is as has been defined.

In particular, we consider the blind signature algorithm provided by Zhang et al. [27] as follows:

(i) (1) Let $k \in \mathbb{Z}_n$ be an integer randomly selected by the signer, which calculates $R = kG$. Then, the signer sends R to the requester.

(ii) (2) Firstly, the requester selects two integers γ and $\delta \in \mathbb{Z}_n$ randomly and computes $A = kG + \gamma G + \delta P = (x, y), t = x \bmod n$. It checks whether t equals zero. If so, the requester reselects γ and δ. Then, it computes $c = SHA256(m\|t)$ and $\hat{c} = c - \delta$; here, SHA256 [29] is a cryptography hash function. Finally, the requester sends $\hat{c}$ to the signer as the blinded message.

(iii) (3) The signer generates a blind signature $\hat{s} = k - \hat{c}d$ using the blinded message and sends to the requester.

(iv) (4) On receiving $\hat{s}$, the requester computes $s = \hat{s} + \gamma$, and along with the above c, the requester gets a signature c, s for the original message m.

(v) (5) Anyone can verify the signature (c, s) by checking the following equation:

$$c = \mathrm{SHA256}\left(m \| R_x(cP + sG) \bmod n\right). \tag{3}$$

Note that here $R_x(cP + sG) \bmod n$ means we get the x resolution values of point $cP + sG$, and || means we concatenate two strings.

3.3. Blockchain. Maintained by many mutual untrusted parties, the leger of a blockchain generally captures the characteristics of decentralization, tamper proof, and traceability. Blockchain technology is an underlying technology of the famous cryptocurrency Bitcoin [30] and has been a prominent development in the past decade. Consequently, many applications are built based on blockchain to acquire the characteristics, so as our proposed voting protocol in this study. The key notations of blockchain technology are as follows:

(i) Ledger. As the name implies, the ledger is used to manage data such as accounts and transaction flow and supports functions such as classified bookkeeping, account reconciliation, and clearing and settlement. In multiparty cooperation, multiple participants hope to jointly maintain and share a timely, correct, and secure distributed ledger to eliminate information asymmetry, improve operational efficiency, and ensure capital and business security. The blockchain is usually regarded as a core technology for building a "distributed shared ledger." Through the joint of a series of technologies such as chained block data structures, multiparty consensus mechanisms, smart contracts, and world state storage, it can achieve a shared ledger that is consistent, credible, transactionally secure, and difficult to tamper with. The basic contents contained in the ledger include blocks, transactions, accounts, and world states.

(ii) Block. Blocks are data structures constructed in chronological order. The new block will introduce the hash information of the previous block and then use the hash algorithm and the data of this block to generate a unique data fingerprint. The sophisticated data structure design makes the data on chain traceable and verifiable.

(iii) Transaction. A transaction can be regarded as a piece of request data sent to the blockchain system, which can be used to deploy contracts, call contract interfaces, maintain the life cycle of contracts, manage assets, and exchange value. The basic data structure of a transaction includes sender, receiver, and transaction data.

(iv) Consensus Mechanism. The consensus mechanism is a core concept in the blockchain. As a distributed system, the blockchain can be jointly calculated by different nodes, which jointly witness the execution process of transactions and confirm the final

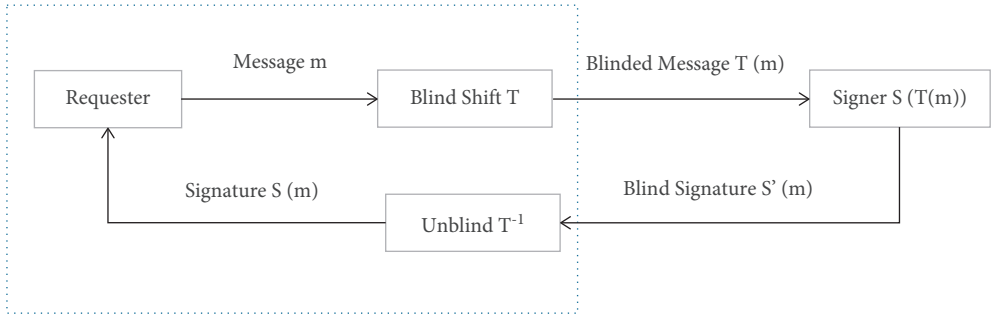

Figure 2: Blind signature scheme.

calculation results. There is a process of cooperating in the blockchain that it can make mutually untrusted participants to reach an agreement and ensure consistency. Continuous cooperation can be abstracted as a "consensus" process. The algorithms and strategies involved are collectively referred to as a consensus mechanism.

(v) Smart Contract. A smart contract refers to a contract defined in digital form that can automatically execute terms. The digital form means that the contract must be implemented in computer code. As long as the parties reach an agreement, the rights and obligations established by the smart contract will be automatically executed. Thus, the result cannot be denied. To run digital smart contracts, the blockchain system must have compilers and executors that can compile, parse, and execute computer code, collectively referred to as a virtual machine system. After the contract is written, it is compiled with a compiler, and a deployment transaction is sent to deploy the contract on the blockchain system. After the deployment transaction consensus is passed, the system assigns a unique address to the contract and saves the binary code of the contract. After the transaction is called, the virtual machine executor loads the code from the contract storage, executes it, and outputs the execution result.

4. Proposed Protocol

In this subsection, we will describe a practical anonymous voting protocol via blockchain. Our scheme has all the aforementioned properties in Section 1.

4.1. System Overview. As shown in Figure 3, the voting protocol consists of four stages: initialization, voting, opening, and verifying/tally. We adopt the ECC system in our scheme, and the elliptic curve is secp256k1. This curve can be described as $T = (p, G, n, a, b, h)$, where a and b are constants, p is the p value of the finite field $F(p)$ of secp256k1, G is the base point, n is the order of G, and h is a cofactor. All these parameters are public.

4.1.1. Initialization. Anyone in the IoE system can launch voting by the proposed IoEPAV. All voters who want to join the voting should provide their public keys and identification. In the initialization stage, all the public information of the voters will be broadcasted to the blockchain through the smart contract in IoEPAV. We assume that there are n_v different voters $v_1, v_2, \ldots, v_{n_v}$. Each voter v_i generates two pairs of ECC keys (sk_{v_i}, pk_{v_i}) and $(\widehat{sk}_{v_i}, \widehat{pk}_{v_i})$. Let addr_{v_i} be the public address of the voter v_i in the Ethereum network, and ID_{v_i} represents the voter's identification. Then, the public information for each voter v_i is a tuple $(ID_{v_i}, pk_{v_i}, \widehat{pk}_{v_i}, \text{addr}_{v_i})$. Everyone can get this information from the blockchain to verify its validity.

4.1.2. Voting. As soon as voting is launched, each voter v_i can start to submit their ballot. Firstly, a cryptography commitment protocol is invoked $\text{Commit} = (C, V)$. Here, we use the algorithm C and the algorithm V will be used later in the final stage. C as $(c_{v_i}, o_{v_i}) \xleftarrow{R} C(m_{v_i})$ is invoked for the ballot message m_{v_i} of voter v_i.

Then, voter v_i generates a $\widetilde{c} = (\widehat{c}_1, \widehat{c}_2, \ldots, \widehat{c}_{n_v})$ and $\widetilde{x} = (\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_{n_v})$ for different voters $v_1, v_2, \ldots, v_{n_v}$, where $v_{n_v} \neq v_i$. Voter v_i completes this by a blind commitment algorithm $(\widetilde{x}, \widetilde{c}) \leftarrow \text{BlindX}(c_{v_i}, \widetilde{pk}_{n_v}, \widetilde{\widetilde{pk}}_{n_v})$ in the blind signature protocol.

Let H_{n_v} be the hash of the tuple $(\text{addr}_i, ID_i, \widehat{c}_{n_v})$ for different voters $v_1, v_2, \ldots, v_{n_v}$, where $v_{n_v} \neq v_i$. Then, voter v_i uses the ECDSA signature algorithm $s_{n_v} = \text{Sign}(sk_i, H_{n_v})$ for other different voters n_v and gets a tuple of signatures $\widetilde{s} = (s_1, s_2, \ldots, s_{n_v})$.

Then, a group information of $(ID_i, \text{addr}_i, \widetilde{c}, \widetilde{s})$ is recorded into the blockchain through the smart contract. Note that $\widetilde{x}$ has been saved in secret by voter v_i himself in this stage. The detailed design of the algorithm C and BlindX is given in Section 4.2.

Once $(ID_i, \text{addr}_i, \widetilde{c}, \widetilde{s})$ generated by voter v_i is recorded, the other voters can generate a blind signature for it. Firstly, every other voter verifies the validity of the signature s_{n_v} by the ECDSA verification algorithm $\text{Verify}(s_{n_v}, pk_i)$. If s_{n_v} is valid, then every other voter generates a blind signature d_{n_v} by a blind signature algorithm $d_{n_v} \leftarrow \text{Blind}S(\widehat{sk}_{n_v}, sk_{n_v})$ in the blind signature protocol and sends d_{n_v} into the blockchain. Let $\widetilde{d} = (d_1, d_2, \ldots, d_{n_v})$ for different voters $v_1, v_2, \ldots, v_{n_v}$ where $v_{n_v} \neq v_i$.

At the end of this stage, we have $(\widetilde{c}, \widetilde{s}, \widetilde{d})$ for a ballot m_{v_i} of voter v_i. The detailed design of the algorithm Blind$S$ will be given in Section 4.2.

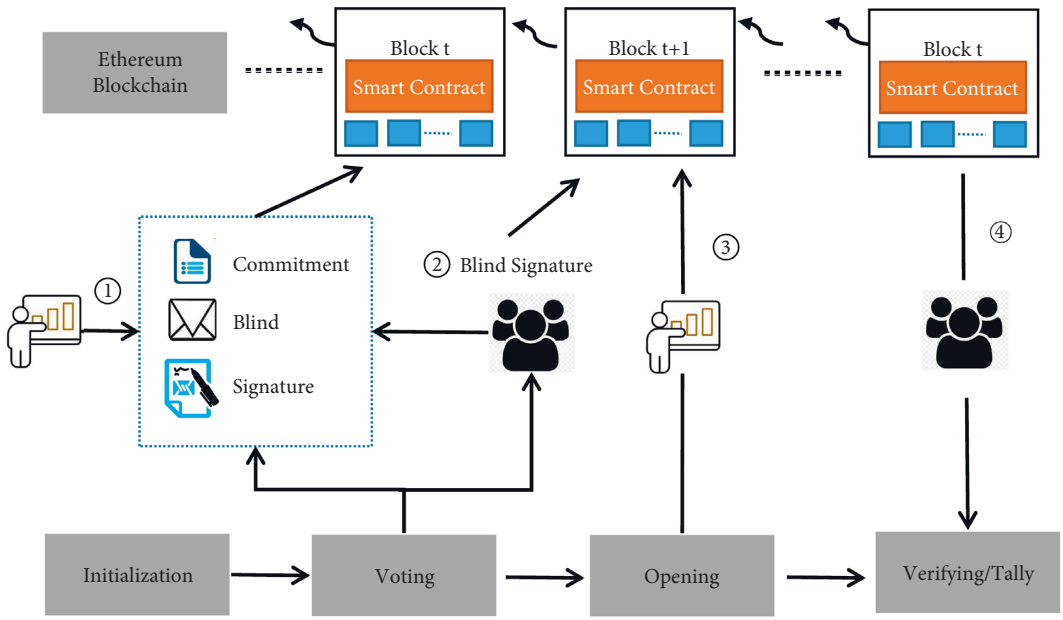

Figure 3: Proposed IoEPAV.

4.1.3. Opening. In this stage, voter v_i will open his/her voting commitment. Firstly, he/she gets the $\tilde{d}$ from the blockchain and calculates the corresponding tuple $\tilde{y} = (y_1, y_2, \ldots, y_{n_v})$ for different voters $v_1, v_2, \ldots, v_{n_v}$ where $v_{n_v} \neq v_i$. Here, y_{n_v} is a signature for the original commitment c_{v_i}. Voter v_i generates $\tilde{y}$ by calculating

$$y_{n_v} = d_{n_v} + \gamma, \tag{4}$$

where γ is a random key saved in Blind$X$.

Given $(x_{n_v}, c_{v_i}, y_{n_v})$, voter v_i can verify the validity of y_{n_v} by an algorithm Verify$S(x_{n_v}, c_{v_i}, y_{n_v}, pk_{n_v})$, where x_{n_v} is saved in Blind$X$. Note that y_{n_v} is a signature of the commitment c_{v_i} from voter v_{n_v}. If the signature y_{n_v} is valid, then voter v_i sends the open commitment string o_{v_i} and x_{n_v} to the blockchain. Note that voter v_i chooses a random address in the Ethereum network to send this transaction. Anyone cannot find out who has sent this transaction. The detailed design of the algorithm Verify$S$ will be given in Section 4.2. Finally, we have $(c_{v_i}, o_{v_i}, \tilde{x}, \tilde{y})$ for an original ballot m_{v_i} of voter v_i in the blockchain.

4.1.4. Verifying/Tally. Now, we have a voting list of $(c_{v_i}, o_{v_i}, \tilde{x}, \tilde{y})$ for each voter v_i. Everyone can use Verify$S$ again to verify the voting. Note that we do not have to verify all the $\tilde{y}$ values for a ballot m_{v_i} of voter v_i. If more than half of $\tilde{y}$ are valid, we think (c_{v_i}, o_{v_i}) is valid. Finally, we can use the $V(c_{v_i}, o_{v_i})$ algorithm in cryptography commitment to open the original ballot m_{v_i} and tally the result of the voting.

4.2. Algorithm Design. In this subsection, we will describe the design of the algorithms mentioned in A in detail. To make the voting service as simple as possible to be integrated into an IoE system, we assume that anyone who tries to use the voting service can ignore the underlying design of smart contracts of the blockchain. The details are as follows:

(i) Cryptography Commitment Algorithms. To construct a cryptography commitment protocol, we should build a pair of efficient algorithms (C, V). We can do this by using a collision-resistant hash function. As shown in Algorithm 1, the algorithm C generates a 32 byte random string as the open string o in the commitment protocol. The commit string c is the hash of the original ballot m and o. Algorithm 2 is a invert process to verify whether (m, o) is corresponded to c. The algorithm C is used at the beginning of the "Voting" stage, while the algorithm V is used at the end of the "Verifying/Tally" stage. We will give the security analysis for this commitment protocol in Section 5.

(ii) Blind Algorithms. We divide the blind signature protocol into three algorithms. They are Blind$X$, Blind$S$, and Verify$S$. As shown in Algorithm 3, Blind$X$ is used for blinding the commit string c_{v_i} for the ballot. γ and δ are two random secret keys. G and n are the public parameters in the ECC and the commit string; the input also includes the public keys of each voter. Following the blind signature protocol mentioned in Section 3, Blind$X$ is used to generate the blinded message $(\tilde{s}, \tilde{c})$. At the end of Blind$X$, each voter keeps the secret data $(x_j, \hat{c}_j, \gamma, \delta)$ and sends $(\text{addr}_{v_i}, \widetilde{\text{addr}}_{n_v}, \tilde{s}, \tilde{c})$ to the blockchain.

Then, in Algorithm 4, Blind$S$ is used for the other voters to sign on $\hat{c}_{n_v}$ submitted by voter v_i in Algorithm 3. With the corresponding secret keys $(\hat{s}k_{n_v}, sk_{n_v})$, each signer can generate a blind signature d_{n_v} as Line 8 in Algorithm 4. Then, voter v_i can easily calculate the explicit signature $y_{n_v} = d_{n_v} + \gamma$ for c_{v_i}. Finally, we have Verify$S$ as shown in

Input (m)
Output (c,o)
(1) o = crypto.randomBytes(32)
(2) c = SHA256(m + serialize(o))
(3) **return** (c, o)

Algorithm 1: Generate commitment.

Input (m, c, o)
Output *True* or *False*
(1) $\hat{c}$ = SHA256(m + serialize(o));
(2) **if** $c \neq \hat{c}$ **then**
(3) **return** *False*
(4) **else**
(5) **return** *True*
(6) **end if**

Algorithm 2: Open commitment.

Input $(c_{v_i}, \widetilde{\widetilde{pk}_{n_v}}, \widetilde{pk}_{n_v})$
(1) γ = crypto.randomBytes(32);
(2) δ = crypto.randomBytes(32);
(3) **for** $j = 0$ to n_v **do**
(4) $A_j = \widehat{pk}_j + \gamma * G + pk_j * \delta$
(5) t_j = getXpointFromPubkey (A_j)mod n
(6) x_j = SHA256($c_{v_i} + t_j$)
(7) $\hat{c}_j = x_j - \delta$
(8) $\tilde{c}$.push$(\hat{c}_j)$
(9) H_i = SHA256 $(ID_i + \text{addr}_i + \hat{c}_j))$
(10) s_j = secp256k1.ecdsaSign(H_i, sk_i)
(11) $\tilde{s}$.push(s_j)
(12) save(x_j, $\hat{c}_j$, γ, δ)
(13) **end for**
(14) await Contract.setAnmVote(addr$_{v_i}$, $\widetilde{\text{addr}}_{n_v}$, $\tilde{s}$, $\tilde{c}$);

Algorithm 3: BlindX.

Algorithm 5 to verify the validity of the signature y_{n_v}. Everyone who gets the public key of the signers from the blockchain can calculate the hash. Here, the blind signature protocol mentioned in Section III is divided into Algorithms 3 to 5.

4.3. Smart Contract Design. In this part, we present the design of the smart contract, which provides interfaces to record the voting data into the blockchain. Thus, the smart contract should involve the necessary data structure referring to the voting scheme. Firstly, we need a data structure to record the information binding with the voter. As shown in Table 2, "address, PK, PKs, and ID" are the basic public information, while the other three mapping data are corresponding to process data generated in the blind signature protocol. Table 3 is designed for storing the results in the "opening" stage. Besides, there are some other variables such as an "unit" for the number of voters. To be succinct, we do not list all of them.

Recall that there are four stages in the whole voting scheme, namely, initialization, voting, opening, and verifying/tally. Then, the smart contract should afford the necessary interfaces for them to interact with the blockchain. The interfaces can be classified into two types: "Write" for recording information into the blockchain and "Query" for querying information from the blockchain. We have 11 interfaces in our smart contract. Considering the space of the paper, only those critical functions are given in detail. However, it is enough for the readers to understand the whole protocol. In the stage "Initialization," a kind of "Write" function is used to record the public identity information of a voter. Then, in the stage "Voting," a "Write" interface named "setAnmVote" is used to record the blind commitment generated in Algorithm 3. As shown in Algorithm 6, *iV* and *iS* are the accounts in the Ethereum blockchain that represents a voter's address and a potential signer's address separately. *isi* and *ici_pi* are data generated in "BlindX." voters is an array corresponding to the data structure in Table 2. The requirement in Line 1 makes sure that only the voter himself can set the data. Once the data have been recorded, no one can reset it including the voter himself.

As soon as the data are confirmed by the blockchain, the other voters acting as a signer will try to generate a blind signature for the commitment. The algorithm for a signer to generate a blind signature is described in Algorithm 4. Firstly, a signer will use "getAnmVote" in Algorithm 7 to query the commitment data generated for him. Anyone can query the commitment according to the public Ethereum account address. In the end, the signer will submit his blind signature idsig through "signAnmVote" in Algorithm 8. Similarly, the requirement in Line 1 makes sure that only the signer himself can set the corresponding data. Once the signature has been recorded, no one can reset it including the signer himself.

The design ideas of the other interfaces are similar to these algorithms given above. When "Write" information to the blockchain, necessary conditions are set. Then, anyone can check the data from the blockchain by the kind of "Query" interface.

5. Security Analysis

In this section, we will discuss why our protocol can resist the potential attacks in the threat models and fulfil all the design goals in Section 3.

5.1. Threat Models. In our scheme, we suppose a voter is a rational one, which means he/she would not let their right to vote become invalid by doing something obviously break the protocol. For instance, a voter should submit his/her blind signature for other voters' ballots correctly; otherwise, his/her right to vote will be thought invalid. Here, we present the threat model specially for a voting service.

Input $(\widehat{sk}_{n_v}, sk_{n_v})$
(1) (v, s_{n_v}, $\widehat{c}_{n_v}$) = await Contract.getAnmtVote (addr$_i$, addr$_{n_v}$)
(2) **if** $v \neq$ addr$_i$ **then**
(3) Return
(4) **end if**
(5) H = SHA256 (ID_i + addr$_i$ + $\widehat{c}_{n_v}$)
(6) S = secp256k1.ecdsaVerify(s_{n_v}, H, pk_i)
(7) **if** S == True **then**
(8) $d_{n_v} = \widehat{sk}_{n_v} - \widehat{c}_{n_v} * sk_{n_v}$
(9) **else**
(10) **Return**
(11) **end if**
(12) await Contract.signAnmVote(addr$_i$, d_{n_v})

Algorithm 4: BlindS.

Input $(x_{n_v}, c_{v_i}, y_{n_v}, pk_{n_v})$
Output *True* or *False*
(1) $B = x_{n_v} * pk_{n_v} + y_{n_v} *$ G
(2) bx = getXpointFromPubkey(B) mod n
(3) H = SHA256(c_{v_i} + bx)
(4) **if** $H == x_{n_v}$ **then**
(5) **Return** *True*
(6) **else**
(7) **Return** *False*
(8) **end if**

Algorithm 5: VerifyS.

Input $(iV, \widetilde{\text{addr}}, \widetilde{s}, \widetilde{c})$
(1) require(msg.sender == iV);
(2) **for** $i = 0$ to n_v **do**
(3) $iS = \widetilde{\text{addr}}$ [i];
(4) $isi = \widetilde{s}$ [i];
(5) $ici\_pi = \widetilde{c}$ [i];
(6) require(voters [iV].si [iS]) == 0);
(7) voters [iV].si [iS] = isi;
(8) voters [iV].ci_pi [iS] = ici_pi;
(9) **end for**

Algorithm 6: setAnmVote.

Table 2: Structure of a voter.

Data type	Description
address	voter
string	PK
string	PKs
string	ID
mapping(address ⇒ string)	si
mapping(address ⇒ string)	ci_pi
mapping(address ⇒ string)	dsigs

Table 3: Structure of an open result.

Data type	Description
string	m
string	bm
string	oi
mapping(address ⇒ string)	ci
mapping(address ⇒ string)	yis
bool	isFinished

(1) Voter Model. Although a voter is a rational one, he/she may try to lead the voting result in his/her favour without breaking the rule of the protocol. First, since the voting is anonymous, a voter may attempt to submit a duplicate ballot to increase his/her chance to vote. Second, because each voter will blindly sign on the blinded ballots, it is possible for a voter to attempt to change the original ballot after the corresponding blind ballot has been blindly signed by others. It means any vulnerability of the blind signature protocol will defeat the whole voting scheme. Third, knowing other voters' public identities, a voter may look for ways to let others' legal ballots become invalid by forging others' ballots.

(2) Adversary Model. An adversary can be anyone who is a user of the IoE system. First, an adversary may attempt to affect other voters' choices through vote buying, voter coercion, and so on. Second, there is a possibility for an adversary to stop an eligible voter from performing the process of the voting protocol. For example, voters can be subjected to DDoS attacks, causing them to malfunction. Third, an adversary may attempt to tamper with the result of the voting.

(3) Blockchain System Model. Attacks against the blockchain system may also cause the failure of the voting scheme, since it is based on the blockchain.

5.2. Cryptography Commitment. A cryptographic commitment scheme Commit = (C, V) is secure when it is both hiding and binding. In our scheme, we constructed the cryptographic commitment using a collision-resistant hash function H (in out construction, we use SHA256).

Input (iV, iS)
Output (V, S, isi, ici_pi)
(1) return (voters $[iV]$.voter, voters $[iV]$.si $[iS]$, users $[iV]$.ci_pi $[iS]$);

Algorithm 7: getAnmVote.

Input (iV, idsig)
(1) require(voters $[iV]$.voter.isvaild() and users $[iV]$.dsigs[msg.sender]).length = = 0);
(2) voters $[iV]$.dsigs[msg.sender] = idsig;

Algorithm 8: signAnmVote.

We now prove that the binding commitment C_H satisfies two properties based on the assumption that H is collision-resistant.

(i) Binding Proof. The binding commitment C_H is a binding commitment if H is collision-resistant. This can be shown immediately as follows: if there exists an adversary A that breaks the binding property, it will immediately give a collision for H. More precisely, for some commitment string c, assume A outputs two pairs (m_1, o_1) and (m_2, o_2), where $m_1 \neq m_2$, but $V(m_1, c, o_1) = v(m_2, c, o_2) = \text{accept}$. Thereafter, we have a collision for H that $H(h_1, o_1) = c = H(m_2, o_2)$. So, we can say that C_H is computationally binding since it depends on a computational assumption for solving this collision for H.

(ii) Hiding Proof. We first consider input hiding required that the distribution $\{H(m_1, o)\}$ is statistically indistinguishable from the distribution $\{H(m_2, o)\}$ for all $m_1, m_2 \in M$, where $o \xleftarrow{r} R$. In our construction, once H is collision-resistant and if the set R is large enough, it is considered input hiding. For example, $R = \{0,1\}^{512}$ should be sufficient for SHA256. This provides a way to build a secure and practical commitment scheme from SHA256. Then, if H is input hiding, no adversary even an unbounded adversary A can break the security of its derived commitment scheme C_H. So, we can say that C_H is unconditionally hiding.

5.3. Blind Signature. The blind signature protocol we used in our scheme is recalled as follows:

(i) (1) The signer n_v randomly generates two pair keys (sk_{n_v}, pk_{n_v}) and $(\widehat{sk}_{n_v}, \widehat{pk}_{n_v})$ and $pk_{n_v}, \widehat{pk}_{n_v}$ are public to the requester v_i.

(ii) (2) The requester selects two integers γ and $\delta \in \mathbb{Z}_n$ randomly and computes $A = \widehat{pk}_{n_v} + \gamma * G + \delta * pk_{n_v} = (xp, yp), t = xp \bmod n$. It checks whether t equals zero. If so, the requester reselects γ and δ. Then, it computes $x = \text{SHA256}(c_{v_i} \| t)$ and $\widehat{c} = x - \delta$, where SHA256 is a hash function with 32 bit words and c_{v_i} is the commitment generated by voter v_i. Finally, the requester sends $\widehat{c}$ to the signer as the blinded message.

(iii) (3) The signer generates a blind signature $d = \widehat{sk}_{n_v} - \widehat{c} * sk_{n_v}$ using the blinded message and sends to the requester.

(iv) (4) On receiving d, the requester computes $y = d + \gamma$, and with its above x, the requester gets a signature x, y for the original message c_{v_i}.

(v) (5) Anyone can verify the signature (x, y) by checking the following equation: $x = \text{SHA256}(c_{v_i} \| R_x(x * pk_{n_v} + yG) \bmod n)$.

Correctness Proof. Firstly, we prove the signature (x, y) to be valid as follows:

$$\begin{aligned} & R_x\left(x * pk_{n_v} + yG\right) \bmod n, \\ & = R_x\left(x * pk_{n_v} + d * G + \gamma * G\right) \bmod n, \\ & = R_x\left(x * pk_{n_v} + \widehat{sk}_{n_v} * G - \widehat{c} * sk_{n_v} * G + \gamma * G\right) \bmod n, \\ & = R_x\left(x * pk_{n_v} + \widehat{sk}_{n_v} * G - (x - \delta) * sk_{n_v} * G + \gamma * G\right) \bmod n, \\ & = R_x\left(\widehat{pk}_{n_v} + \gamma * G + \delta * pk_{n_v}\right) \bmod n, \\ & = R_x(xp, yp) \bmod n, \\ & = t. \end{aligned} \tag{5}$$

It follows that $x = \text{SHA256}(c_{v_i} \| R_x(x * pk_{n_v} + yG) \bmod n)$, which means that (x, y) is a valid signature of c_{v_i}.

Blindness Proof. Secondly, we show the blindness of the protocol. We define view $\mathbb{V}$ for a signer during the process of the protocol. For example, let (x, y) be the signature of c_{v_i} that has been generated in the protocol. Then, view $\mathbb{V}$ consists of sk_{n_v}, $pk_{n_v} = sk_{n_v} * G$, $\widehat{sk}_{n_v}$, $\widehat{pk}_{n_v} = \widehat{sk}_{n_v} * G$, $\widehat{c}$, and $d = \widehat{sk}_{n_v} - \widehat{c} * sk_{n_v}$. We then show that for any given view $\mathbb{V}$ and valid message signature pair $(c_{v_i}, (x, y))$, blinding factors γ and δ exist and are unique. Then, for γ and δ, we have

$$\begin{aligned}\widehat{c} &= (x-\delta)\mathrm{mod}n,\\ \gamma &= \left(y-\widehat{sk}_{n_v}+\widehat{c}*sk_{n_v}\right)\mathrm{mod}n,\\ x &= \mathrm{SHA256}\left(c_{v_i}\middle\|R_x\left(\widehat{pk}_{n_v}+\gamma*G+\delta*pk_{n_v}\right)\mathrm{mod}n\right).\end{aligned} \tag{6}$$

Then, $R_x(\widehat{pk}_{n_v}+\gamma*G+\delta*pk_{n_v})\mathrm{mod}n$ is uniquely determined by x and c_{v_i}. To make it succinct, we note $\widehat{sk}_{n_v}+\gamma+\delta*sk_{n_v}=R_x^{-1}(c_{v_i}\mathrm{°x})$, which means $\widehat{sk}_{n_v}+\gamma*G+\delta*sk_{n_v}$ is uniquely determined by x and c_{v_i}.

Then, $\delta*sk_{n_v}=(R_x^{-1}(c_{v_i}\mathrm{°x})-\widehat{sk}_{n_v}-\gamma)\mathrm{mod}n=(R_x^{-1}(c_{v_i}\mathrm{°x})-\widehat{\mathrm{sk}}_{\mathrm{n_v}}-\mathrm{y}+\widehat{\mathrm{sk}}_{\mathrm{n_v}}-\widehat{c}*\mathrm{sk}_{\mathrm{n_v}})\mathrm{modn}=(\mathrm{R}_{\mathrm{x}}^{-1}(\mathrm{c}_{\mathrm{v_i}}\mathrm{°x})-\mathrm{y}-\widehat{c}*sk_{n_v})\mathrm{mod}n$.

Finally, we have

$$\begin{aligned}\gamma &= \left(y-\widehat{sk}_{n_v}+\widehat{c}*sk_{n_v}\right)\mathrm{mod}n,\\ \delta*sk_{n_v} &= \left(R_x^{-1}\left(c_{v_i}\mathrm{°x}\right)-y-\widehat{c}*sk_{n_v}\right)\mathrm{mod}n.\end{aligned} \tag{7}$$

Then, γ and δ can be uniquely determined by $(x, y, c_{v_i}, \widehat{sk}_{n_v}, sk_{n_v}, \widehat{c})$. All $(x, y, c_{v_i}, \widehat{sk}_{n_v}, sk_{n_v}, \widehat{c})$ are in the view $\mathbb{V}$.

5.4. Security Properties. First, we will show how the protocol fulfils the seven design goals.

Fairness. The proposed voting scheme is a fair voting by succeeding the hiding property of the cryptography commitment protocol. Since we have proven the security of the cryptography commitment protocol used in our scheme, no one can obtain the ballot results of others before he/she has submitted his/her ballot.

Decentralization. Directly, the protocol is decentralized as it is built without any TTP. We use smart contracts to automate the voting process of the Internet of energy. Eligibility. In the initialization stage, we check all the voters' identities and public information to make sure the eligibility.

Anonymity. The correctness and blindness of the blind signature protocol make sure that no one can know the owner of a ballot from the voting result at the end. Since we have proven the security of the blind signature protocol used in our scheme, the privacy of voters can be protected.

Compatibility. As the protocol is realized by smart contract and Web3 [31], it can be integrated into an IoE system easily. Any blockchain system including Ethereum 2.0, which supports smart contracts, is feasible for the proposed scheme and we do not need to construct a whole new blockchain platform.

Verifiability. The data generated in each stage of the voting can be checked from the blockchain. Therefore, the voting is verifiable.

Coercion Resistance. Even though one tries to compel a voter to vote by his/her instruction, he/she cannot find out whether the coerced voter has done as he/she wishes.

5.5. Resistance against the Threat Model. Finally, we will show how the protocol resists the potential attacks in the threat models. (1) Resistance against Voter: first, a voter cannot submit a duplicate ballot because the design of the smart contract will reject a piece of duplicate information. Second, a voter cannot change the original ballot for the security of the blind signature protocol. Third, to forge others' ballots, a voter should get their secret keys or break the ECC asymmetric cryptography. (2) Resistance against Adversary: first, since the voting is anonymous, an adversary cannot affect other voters' choices. Second, to stop an eligible voter from voting, an adversary should break the security of the blockchain. The blockchain makes sure that the result of the voting cannot be tampered with.

(3) Resistance against Blockchain: because our protocol is designed via the Ethereum blockchain, which is proven secure in practice, and thousands of applications have been built on it. A 51% attack is still hypothetical by a group of miners controlling more than 50% of the network's mining hash rate or computing power. For voting, if more than 50% of voters collude, it is not necessary to launch voting. Besides, the fully decentralized architecture of blockchains makes them robust against DoS/DDoS attacks.

6. Implementation and Performance

6.1. Implementation Description. We use a PC with an OS version of Ubuntu 18.04 64x as a user client. The CPU and memory are Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz 2.30 GHz and 8G separately. We implemented our protocol in two parts, namely, Web3 programs and smart contracts. We write the smart contract in Solidity. We use JavaScript to finish the Web3 programs along with several libraries. "Ethers.js" is an Ethereum library to interact with the Ethereum blockchain. "Crypto.js" is a JavaScript library to realize the cryptography protocol adopted in our voting protocol. The experiment code is available at https://github.com/researchSec/IoEPAV. As shown in Figure 4, each user plays as a voter and a blind signer simultaneously. They interact with the blockchain network through the Web3 application. Indeed, the Web3 application will invoke the smart contract to read or write voting data on the blockchain network. Note that the users do not need to involve in the mining or consensus progress of the blockchain network. For simplicity, we use multiple Ethereum accounts to represent different voters in the Web3 application to invoke the smart contract. To evaluate the performance in a more reliable and practical manner, we deploy the smart contract on two popular test networks of Ethereum and have strong links to the main network. Besides, we build an Ethereum development with a professional tool called Hardhat. Thus, in the connectivity model, we run our test cases in three blockchain networks separately. They are Hardhat local network, Rinkeby test network, and Ropsten test network.

The Ethereum network that deals with real money is called "mainnet," and then, other live networks named "testnets" (multiple ones) are also provided by Ethereum. In testnets, the network does not deal with real money but does mimic the real-world scenario well. Ropsten and Rinkeby are the testnets we choose. Ropsten is a proof of work (PoW) testnet. This means it is the best like-for-like representation of Ethereum. Rinkeby is proof of authority (PoA) [32] testnet. In Table 4, we give the state of the two blockchain networks when we performed the experiment. In the testnet

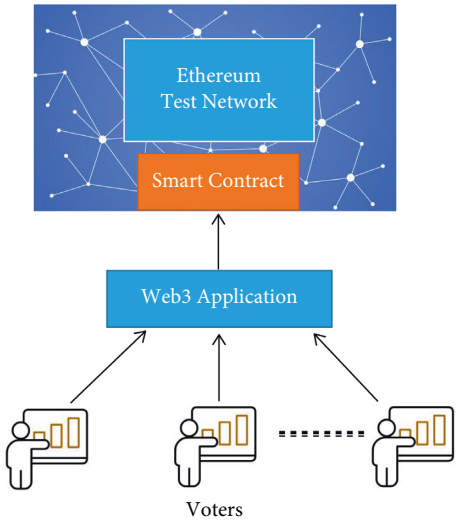

Figure 4: Connectivity model.

Table 4: State of the blockchain Network.

Test network	Consensus	Average block time (S)	Active nodes/ miners	Latest block number
Rinkeby	PoW	47.18	42	10027471
Ropsten	PoA	15.03	44	11869370

Ropsten, the coin is mined following the same scheme as the mainnet. Rinkeby uses PoA consensus schemes, which are a potential direction of Ethereum evolutions.

6.2. Performance in Local Environment. Firstly, we test the key steps in the local development environment to see the performance without considering the latency and throughput according to the network condition. As shown in Table 5, the time for each stage of the voting is proportionate to the increased number of voters. The result is quite straightforward to understand since more voters mean more commitments to be generated. However, even if the number of voters is up to 50, the time is not more than 2 seconds. The time for verifying does not exceed one second. Thus, one can launch voting efficiently.

6.3. Performance in Live Networks. The performance of the voting scheme is relative to the network condition of the live networks. To give an overall evaluation of the system, we give the average response time, transaction cost, throughput, and latency for the execution of the smart contract. To make it clear, we give the description of evaluation properties as follows.

Average Response Time. This metric indicates the time taken to send a transaction to the blockchain and get a response. Note that when a user gets a response, it does not mean that the sent transaction has been confirmed by the nodes/miners of the blockchain.

Transaction Cost. Gas cost is the transaction cost. Gas refers to the fee that is required to successfully execute a contract on the Ethereum blockchain platform. We use the base unit "Gas." 4×10^{-9} Ether $\approx$ 1 Gas.

Transaction Acceptance Latency (TAL). Firstly, a signed transaction is created. The user sends the created transaction to an Ethereum network and captures the current time point T_s. When the transaction is confirmed, the user captures the current time T_e. This metric indicates the time of $t = T_e - T_s$.

Throughput. We consider two metrics, namely, (a) read throughput called QPS (query rate per second) and (2) transaction throughput called TPS (transactions per second). QPS indicates the total number of reading transactions performed within a defined period of time. TPS indicates the ratio of valid transactions that are initiated within a defined period of time.

Firstly, we give the response time of each stage in the voting scheme on different test networks. As shown in Figures 5–9, the performances are quite stable in all live networks. The response time for each stage of the voting is proportionate to the increased number of voters. The result is quite straightforward to understand since more voters mean more commitments to be generated. The performances in the two test networks are quite similar.

To evaluate how the network conditions affect the performance of the voting scheme, we give the transaction acceptance latency (TAL) and throughput in Figures 10 and 11. The TPS and QPS are quite similar between Rinkeby and Ropsten. This can be an explanation for the result of the response time above. Since the throughput in the mainnet of the Ethereum blockchain is about 85 times the size of that in the test networks, we can infer that the performance of the voting scheme will act more efficiently in the mainnet. Note that the TAL varies significantly with different loads (different numbers of concurrent transactions). In the experiment, we give the number of concurrent transactions according to the number of voters.

In Table 6, we also give the average transaction fee for the smart contract deployment and each stage of the voting scheme. As we can see, the gas cost in the two test networks has little difference since the size of the smart contract and transactions are identical.

6.4. Cost Comparison. In Yang's blockchain-based scheme [16], basic cryptographic operations are also introduced. The difference is we use scalar multiplication on an elliptic curve, while Yang uses exponentiation in a multiplicative group. However, if we only care about computational complexity, it is similar.

(i) In ECC, given an elliptic curve of size n, the number of double-and-add steps is proportional to $O(k)$ for $k * G$. Each double/add is a sequence of a constant number of field multiplications, squares, additions, and subtractions. Multiplication and squares are the expensive ones, and using the Karatsuba algorithm as mentioned in [33], they are $O(n^{1.58})$. Therefore, the result is $O(kn^{1.58})$.

(ii) In comparison, the computational complexity of a modular exponentiation of form $g^a \bmod b$ is similar.

Table 5: Average response time.

Number of voters	Initialization (MS)	Blind messages (MS)	Blind signature (MS)	Opening (MS)	Verifying (MS)
5	374.0	323.43	212.0	282.43	109.43
10	635.85	499.14	436.43	561.43	149.43
30	1314.14	1298.71	1234.71	1129.29	228.14
50	2162.28	2314.43	2271.14	1897.43	383.86

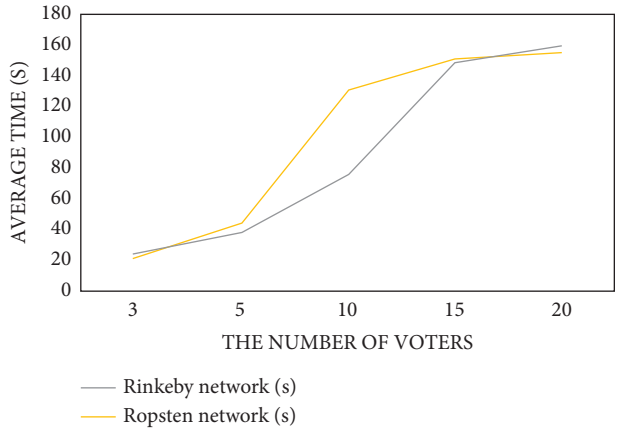

Figure 5: Initialization.

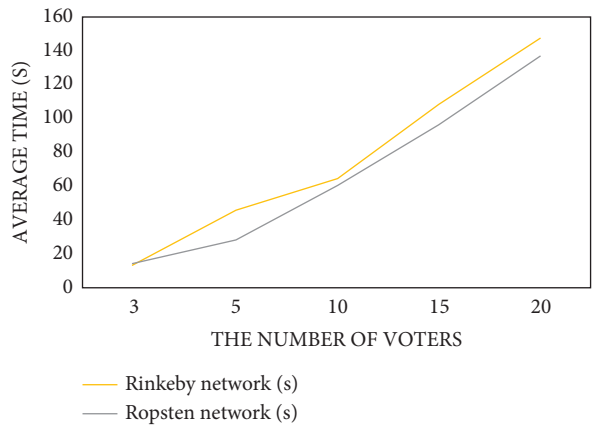

Figure 6: Blind messages.

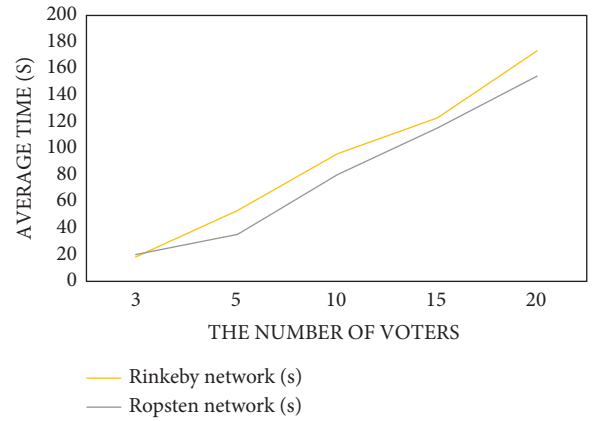

Figure 7: Blind signature.

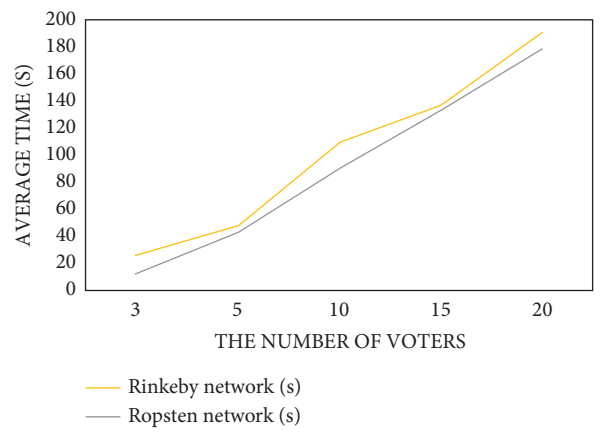

Figure 8: Opening.

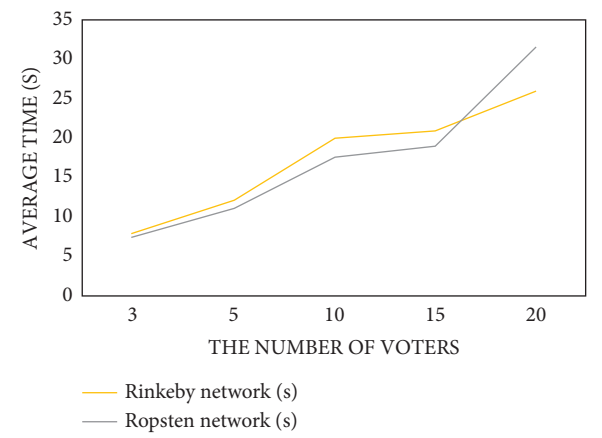

Figure 9: Verifying.

Figure 10: Throughput.

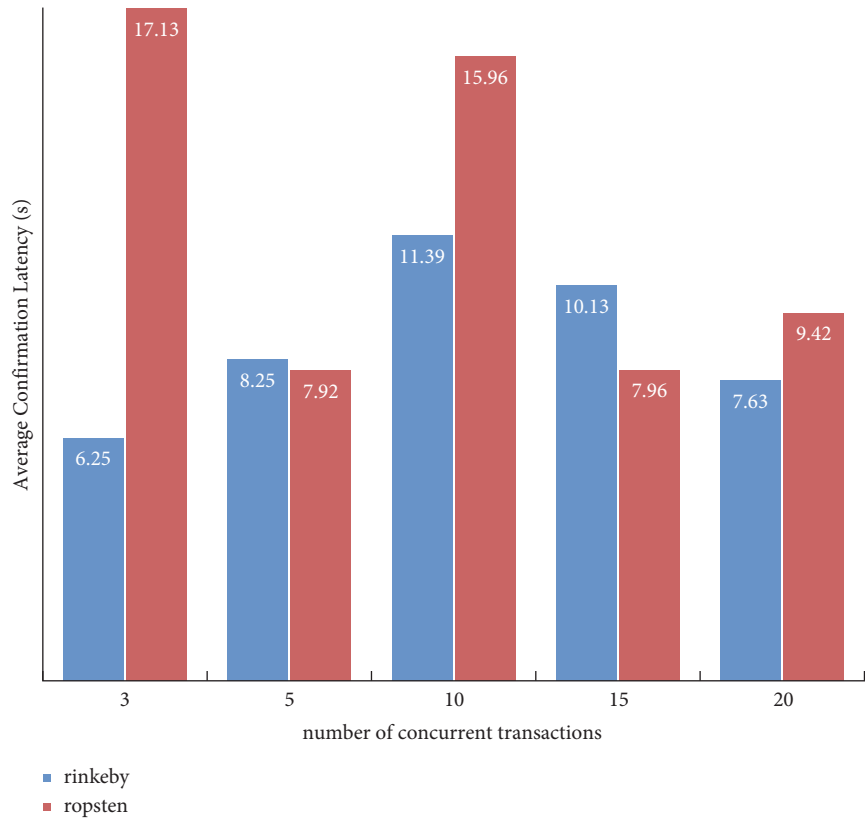

Figure 11: Transaction acceptance latency.

Table 6: Average transaction cost.

Test network	Deployment (Gas)	Initialization (Gas)	Blind messages (Gas)	Blind signature (Gas)	Opening (Gas)
Rinkeby	2,353,838	251,023	204,768	92,811	289,189
Ropsten	2,353,838	251,035	204,780	92,823	284,258

Table 7: Count of the cryptographic operations.

Protocol	Voting	Verifying/tally
IoEPAV	$6 * t * n_v$	$2 * t * n_v$
Yang [16]	$(5 * n_v * t + 3 * t) * n_v$	$(5 * n_v * t + 5 * t) * n_v * n_v$

Square-and-multiply is $O(a)$. Each square/multiply is $O(b^{1.58})$. Therefore, the result is $O(ab^{1.58})$.

According to Yang's performance analysis, only the most time-consuming operations are taken into account. t is denoted as the time of one exponentiation calculation such as a $g^a \bmod b$. Correspondingly, we use t to denote the time of one multiplication calculation in ECC such as $k * G$. Let t_E and t_D be the time of encryption and decryption separately. Then, we let t_S and t_V be the time of executing ECDSA signature and verification, respectively. Approximately, $t_E = 2t$, $t_D = t$, $t_S = t$, and $t_V = t$. n_v is used to denote the number of voters. In Yang's scheme, n_c is the number of candidates, respectively. Note that we can take over the blind signer to candidates in our protocol. To be succinct for comparison, we let $p = n_c = n_v$. The cost of these operations in Yang's and our scheme for comparison is given in Table 7. As shown in Table 7, our scheme IoEPAV is at a lower cost.

7. Conclusion and Future Work

We present a novel blockchain-based voting scheme for IoE system. By getting rid of a trusted third party, the proposed scheme can avoid the single point of failure and is available for a trustless environment. In the past proposals, it is difficult to capture all the required features for a voting scheme. To the best of our knowledge, our scheme is the first one to fulfil all the design goals simultaneously. To achieve this, we combine the cryptography commitment and blind signature protocol. We also use smart contracts to automate the voting process of the Internet of energy. With smart contracts, the voting scheme can be easy to integrate into the IoE system. A voter can follow the voting protocol by invoking the interfaces of the smart contracts. Although we do not use any high-performance cryptography library, the performance in a real environment demonstrates the feasibility of our protocol.

In the future, we can try to use parallel computation in the voting stage to improve efficiency as well. Another

interesting open problem is to create a version of the voting scheme that reduces the number of blind signatures. Making the number of blind signatures irrelative to the number of voters will be a great improvement. This would give a solution that is more efficient with large-scale voters.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by National Key Research and Development Program of China (grant no. 2019YFE0118700), Science and Technology Project of China Southern Power Grid Corporation (grant no. 066600KK52200016).

References

[1] X. Li, M. Ye, J. Chen, J. Chen, and Y.-C. Chen, "A Novel Hierarchical Key Assignment Scheme for Data Access Control in Iot," *Security and Communication Networks*, vol. 2021, Article ID 6174506, 12 pages, 2021.

[2] B. Xu, Q. Chen, J. Ma, Yu Yan, and Z. Zhang, "Research on a kind of ubiquitous power internet of things system for strong smart power grid," in *Proceedings of the IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, pp. 2805–2808, IEEE, Chengdu, China, May 2019.

[3] Internet of energy for electric mobility home page, http://www.artemis-ioe.eu/ioe_project.htm, 2021.

[4] Y. R. Kafle, K. Mahmud, S. Morsalin, and G. E. Town, "Towards an internet of energy," in *Proceedings of the IEEE International Conference on Power System Technology*, September 2016.

[5] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: a survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[6] J. Yin, Y. Xiao, Q. Pei et al., "Smartdid: a novel privacy-preserving identity based on blockchain for iot," *IEEE Internet of Things Journal*, p. 1, 2022.

[7] T. M. Harrison, T. A. Pardo, and M. Cook, "Creating open government ecosystems: a research and development agenda," *Future Internet*, vol. 4, no. 4, pp. 900–928, 2012.

[8] D. Boneh and V. Shoup, "A graduate course in applied cryptography," 2017, https://crypto.stanford.edu/dabo/cryptobook/BonehShoup04.pdf.

[9] K.-H. Wang, S. K. Mondal, Ki Chan, and X. Xie, "A review of contemporary e-voting: requirements, technology, systems and usability," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 31–47, 2017.

[10] A. Ben, "Helios: web-based open-audit voting," *USENIX security symposium*, vol. 17, pp. 335–348, 2008.

[11] M. Backes, M. Gagné, and M. Skoruppa, "Using mobile device communication to strengthen e-voting protocols," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, pp. 237–242, New York, NY, USA, November 2013.

[12] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," pp. 244–251, Springer, Berlin, Germany, 1992.

[13] M. Amine Ferrag and L. Shu, "The Performance Evaluation of Blockchain-Based Security and Privacy Systems for the Internet of Things: A Tutorial," *IEEE Internet of Things Journal*, vol. 8, no. 24, 2021.

[14] FD Team, "Introducing a secure and transparent online voting solution for modern age: follow my vote," https://followmyvote.com.

[15] N. Gailly, P. Jovanovic, B. Ford, J. Lukasiewicz, and L. Gammar, "Agora: Bringing Our Voting Systems into the 21st century," 2018, https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf.

[16] X. Yang, X. Yi, S. Nepal, A. Kelarev, and F. Han, "Blockchain voting: publicly verifiable online voting protocol without trusted tallying authorities," *Future Generation Computer Systems*, vol. 112, pp. 859–874, 2020.

[17] K. Peng, M. Li, H. Huang, C. Wang, S. Wan, and K. K. R Choo, "Security challenges and opportunities for smart contracts in internet of things: a survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12004–12020, 2021.

[18] Q. Pei, E. Zhou, Y. Xiao, D. Zhang, and D. Zhao, "An efficient query scheme for hybrid storage blockchains based on merkle semantic trie," in *Proceedings of the 2020 International Symposium on Reliable Distributed Systems (SRDS)*, pp. 51–60, IEEE, Shanghai, China, September 2020.

[19] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto, *An improvement on a practical secret voting scheme*, pp. 225–234, Springer, Berlin, Germany, 1999.

[20] P. Grontas, A. Pagourtzis, and A. Zacharakis, "Coercion resistance in a practical secret voting scheme for large scale elections," in *Proceedings of the 201714th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, pp. 514–519, IEEE, Exeter, UK, June 2017.

[21] G. Srivastava, A. Dhar Dwivedi, and R. Singh, "Phantom protocol as the new crypto-democracy," in *Proceedings of the 2018IFIP International Conference on Computer Information Systems and Industrial Management*, pp. 499–509, Springer, Olomouc, Czech Republic, September 2018.

[22] C. Braghin, S. Cimato, S. Raimondi Cominesi, E. Damiani, and L. Mauri, "Towards blockchain-based e-voting systems," in *Proceedings of the 2019 International Conference on Business Information Systems*, pp. 274–286, Chittagong, Bangladesh, October 2019.

[23] Li Peng and J. Lai, "Lat-voting: traceable anonymous e-voting on blockchain," in *Proceedings of the 2019 International Conference on Network and System Security*, pp. 234–254, December 2019.

[24] M. Naor, "Bit commitment using pseudorandomness," *Journal of Cryptology*, vol. 4, no. 2, pp. 151–158, 1991.

[25] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, pp. 199–203, Springer, Berline, Germany, 1983.

[26] Q. C. ShenTu and J. P. Yu, "A Blind-Mixing Scheme for Bitcoin Based on an Elliptic Curve Cryptography Blind Digital Signature Algorithm," 2015, https://arxiv.org/abs/1510.05833.

[27] F. Zhang, C. Wang, and Y. Wang, "Digital signature and blind signature based on elliptic curve," *JOURNAL-CHINA*

INSTITUTE OF COMMUNICATIONS, vol. 22, no. 8, pp. 22–28, 2001.

[28] R. Shaikh, M. Nenova, G. Iliev, and Z. Valkova-Jarvis, "Analysis of standard elliptic curves for the implementation of elliptic curve cryptography in resource-constrained e-commerce applications," in *Proceedings of the 2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, pp. 1–4, IEEE, Tel-Aviv, Israel, November 2017.

[29] D. Rachmawati, J. T. Tarigan, and A. B. C. Ginting, "A comparative study of message digest 5 (md5) and sha256 algorithm," in *Journal of Physics: Conference Series*vol. 978, IOP Publishing, Article ID 012116, 2018.

[30] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[31] W.-M. Lee, "Using the web3. js apis," in *Beginning Ethereum Smart Contracts Programming*, pp. 169–198, Springer, Berlin, Germany, 2019.

[32] O. Hasan, L. Brunie, and E. Bertino, "Privacy Preserving Reputation Systems Based on Blockchain and Other Cryptographic Building Blocks: A Survey," University of Lyon, INSA-Lyon, CNRS, LIRIS, UMR5205, Lyon, France, 03034994, 2020.

[33] D. J. Bernstein, C. Chuengsatiansup, and T. Lange, "Curve41417: Karatsuba revisited," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 316–334, Springer, Berlin, Germany, 2014.