A Blockchain-Based Self-Tallying Voting Protocol in Decentralized IoT

Yannan Li, *Student Member, IEEE*, Willy Susilo, *Senior Member, IEEE*, Guomin Yang, *Senior Member, IEEE*, Yong Yu, *Member, IEEE*, Dongxi Liu, Xiaojiang Du, *Fellow, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

Abstract—The Internet of Things (IoT) is experiencing explosive growth and has gained extensive attention from academia and industry in recent years. However, most of the existing IoT infrastructures are centralized, which may cause the issues of unscalability and single-point-of-failure. Consequently, decentralized IoT has been proposed by taking advantage of the emerging technology called blockchain. Voting systems are widely adopted in IoT, for example a leader election in wireless sensor networks. Self-tallying voting systems are alternatives to unsuitable, traditional centralized voting systems in decentralized IoT. Unfortunately, self-tallying voting systems inherently suffer from fairness issues, such as adaptive and abortive issues caused by malicious voters. To address these issues, in this article, we introduce a framework of the self-tallying voting system in decentralized IoT based on blockchain. We propose a concrete construction and prove that the proposed system satisfies all the security requirements, including fairness, dispute-freeness, and maximal ballot secrecy. We simulate the algorithms on a laptop, an Android phone, and a Raspberry Pi to test the time consumption and evaluate the gas cost of each algorithm in a private blockchain as well. The implementation results demonstrate the practicability of our system.

Index Terms—Internet-of-things, E-voting, self-tallying, blockchain, zero-knowledge proof

1 Introduction

The Internet of Things (IoT) is a system comprised of smart devices, actuators, sensors and other objects that are connected throughout the network with the ability to transfer data, share resources and make decisions without man-to-man or man-to-device interaction. IoT has gained extensive attention in industrial communities and the IoT market is expected to reach $500 billion by 2020.[1] Organizations in various industries utilize IoT for better efficiency, convenience and service.[2] Besides the well-known applications of smart cities and smart homes, IoT has potential in many other public and private applications, such as manufacturing, agriculture, transportation, and healthcare. In recent years, some new extensions of IoT are proposed catering to specific needs in different scenarios, such as IIoT and NB-IoT.

Most of the IoT implementations are with centralized infrastructure. Specifically, the devices are linked to the cloud, controlled by a central hub and communicated by C/S models, which is subject to several issues. First, all devices in the system are identified and authenticated by the central server, which requires a huge processing capacity. Second, centralization induces irrational use of resources, since the connections and communications among devices are exclusively through the server, even if they are close to one another. Third, centralized frameworks suffer single-point-of-failure issues.

To overcome the bottleneck of the centralized framework in IoT, the notion of decentralized IoT was proposed. A decentralized paradigm of IoT is promising to solve many issues of a centralized IoT. However, establishing such a framework is quite challenging.[3] Blockchain is an emerging technology, which is a public ledger that achieves decentralization through cryptographic tools and consensus. With blockchain, communications between machines and sensors become easy and effortless. Due to the desirable features, blockchain has many impressive applications [41], [42], [43], [44], [45]. A majority of decentralized IoT leverages Blockchain [47] to build the underlying P2P network. A San

1. Driving Unconventional Growth Through the Industrial Internet of Things. https://www.accenture.com/t20150523T023633Z__w__/us-en/_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Dualpub_11/Accenture-Driving-Unconventional-Growth-through-IIoT.pdfla=en

2. https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

3. https://techcrunch.com/2016/06/28/decentralizing-iot-networks-through-blockchain/

- *Y. Li, W. Susilo, and G. Yang are with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mail: yl738@uowmail.edu.au, {wsusilo, gyang}@uow.edu.au.*
- *Y. Yu is with the School of Computer Science, Shaanxi Normal University, Xi'an, Shaanxi 710062, China. E-mail: yuyong@snnu.edu.cn.*
- *D. Liu is with Data61, CSIRO, Marsfield, NSW 2122, Australia. E-mail: Dongxi.Liu@data61.csiro.au.*
- *X. Du is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA. E-mail: dxj@ieee.org.*
- *M. Guizani is with the Department of Computer Science and Engineering, Qatar University, Doha 2713, Qatar. E-mail: mguizani@ieee.org.*

Manuscript received 20 Sept. 2019; revised 1 Mar. 2020; accepted 3 Mar. 2020. Date of publication 10 Mar. 2020; date of current version 17 Jan. 2022. (Corresponding authors: Yannan Li and Yong Yu.) Digital Object Identifier no. 10.1109/TDSC.2020.2979856

Francisco startup Helium[4] has built a blockchain-based machine network for IoT.[5]

Voting Systems and Decentralized IoT. Voting systems have wide applications in IoT. Two typical examples are provided here. 1) Leader election in decentralized IoT. Leader elections are one of the most common and important activities in a decentralized IoT, such as wireless sensor networks [3], [4]. The goal of a leader election is to designate a special node as an organizer to coordinate tasks in distributed nodes, breaking the inner symmetry in distributed systems. The peers in the network communicate among themselves to vote for a leader. 2) Decision making in IoT systems. One of the most salient features of IoT systems is to collect data and make smarter decisions via voting accordingly.[6] To measure the data of the surrounding environment, such as the temperature, density, etc, in industrial processes, redundant sensors will be deployed. Devices measure various types of data and leverage diverse methods to analyze data, which may lead to a different opinion to a specific decision. Then, devices vote for a final decision. Take the environmental health IoT as an example, which comprises some smart phones with apps to acquire the environmental parameters, including temperature, humidity, noise, and dust, with high accuracy. All parameters are closely related to people's health. Thus, environmental health IoT is an important reference for healthcare. The smart devices within an environmental health IoT collaborate to make decisions to check whether the current environment is suitable to live or work in.

Traditional voting systems with a central party organizing the voting activities are unsuitable for a decentralized IoT framework. As an alternative, self-tallying schemes were proposed, which does not need a third party to tally the ballots and reveal the final result. Instead, after all the voters cast the ballots, anyone can collect the ballots and compute the final results simultaneously. However, self-tallying schemes inherently suffer from fairness issues, in the sense that a malicious voter (sensor) can collect other voters' ballots to compute the final result before casting his/her own ballot. That is, they can know the final result ahead of schedule. Moreover, a voter may refuse to reveal his ballot, making it hard to obtain the final result due to an abortive issue.

Our Contributions. As a consequence, in this paper, we aim to improve the fairness of blockchain-based self-tallying systems for decentralized IoT. The contributions of this paper are listed as follows.

1) We formalize the system model of self-tallying voting systems based on blockchain in decentralized IoT.
2) We propose a concrete construction of a blockchain-based self-tallying voting protocol in decentralized IoT, and prove that it satisfies fairness, dispute-freeness, and maximal ballot secrecy. Specifically, in our construction, we modify the commitment in [22] and the recovery phase in [21] to handle abortive issues, and suggest that using timed commitment to deal with adaptive issues in self-tallying voting schemes.
3) We implement the proposed protocol on a laptop, a mobile phone and Raspberry Pi respectively to test the time consumption. The gas cost is also evaluated on a private blockchain. The implementation results demonstrate its practicality in real-world applications.

Organization. The rest of the paper is organized as follows. We review the related work and provide some preliminaries in Sections 2 and 3, respectively. The system and security models are presented in Section 4. We build a fair blockchain-based self-tallying voting system for decentralized IoT in Section 5 and the security proofs are provided in Section 6. The performance of the proposed protocol is illustrated in Section 7. Finally, we conclude the paper in Section 8.

2 Related Work

Blockchain-Based IoT Solutions. Resource constraint, storage limitation and security are the main hindrances for IoT systems. Researchers and companies have explored the potential of blockchain in IoT systems, with the topics focusing on different aspects of IoT, such as device management, access control, supply chain, IoT security review and so on [5], [6], [7], [8]. Several solutions require additional off-chain storage [9], [10]. Some of the solutions integrate cloud [11], [12] to the blockchain-based IoT, in which blockchain is the overlay of the systems but they are not fully decentralized [9], [14]. Some solutions leverage private blockchains and eliminate the proof-of-work [9], [13], [15]. Aiming to IoT, [16] presented a blockchain platform for IIoT based on the distributed app (DApp), which could be applicable to industrial and manufacturing applications. Slock,[7] a German startup, uses smart contracts to manage the lock of real-world property and achieves fair exchange between users directly. More potential solutions to IoT issues of blockchain can be found in [17].

Self-Tallying E-Voting. E-voting is a flourishing and fadeless topic in academic research. In traditional centralized e-voting protocols, a central authority is usually involved in organizing the election and counting the votes. To achieve stronger voter privacy, Kiayias and Yung [18] proposed the notion of self-tallying voting, which is a new paradigm in decentralized e-voting systems. In self-tallying systems, tallying is an open procedure in which any party, including voters and observers, can validate of each ballot and compute the final voting result after collecting all the valid ballots. They proposed the first concrete construction by leveraging a bulletin board, which achieves perfect ballot privacy and dispute-freeness, but the computational cost is linear with the number of voters. Groth *et al.* [19] proposed a simpler scheme with better efficiency for each voter. They also constructed an anonymous broadcast channel with perfect message secrecy at the cost of increased round complexity of the protocol, which needs $n+1$ rounds for n voters. Hao *et al.* [20] proposed a self-tallying voting protocol based on a two-round anonymous veto protocol (AV-net). Their protocol provides the same security properties and achieves better efficiency in terms of round complexity. Khader *et al.* [21]

4. https://www.helium.com/
5. https://internetofbusiness.com/helium-blockchain-machine-network-iot-unleashed/
6. http://healthandlearning.org/wp-content/uploads/2017/04/Decision-Making-Models-Voting-versus-Consensus.pdf
7. Slock.it. https://slock.it.

claimed that [20] is neither robust nor fair, and advanced the protocol by adding a commitment phase and a recovery round. However, the commitment phase is expensive and the recovery phase ignores the ballots of the abortive voters in their construction.

Blockchain-Based e-Voting Systems. There are already some existing works on blockchain-based e-voting protocols. The role of blockchain in e-voting protocols varies from scheme to scheme. Most of the works incorporate blockchain with bulletin boards and still employ a trusted authority for voter privacy, such as Follow My Vote[8] and TIVI[9,10]. Some of the existing works are based on cryptocurrencies, such as Bitcoin [23], [24] and privacy-enhacing altcoins [25]. Takabatake *et al.* [34] proposed a voting protocol based on Zerocoin to enhance voter privacy. In 2017, McCorry *et al.* [22] presented Open Vote Network,[11,12] the first implementation of a decentralized self-tallying e-voting protocol based on blockchain. The commitment in [22] is the hash of the vote, which is irrecoverable if a voter refuses to cast his ballot in the voting phase. Netvote[13] is a decentralized voting platform on Ethereum. The users can download the DApp to interact with the system in order to vote.

3 Preliminaries

In this section, we provide some preliminaries used in our construction.

3.1 Intractable Assumptions

1) *Discrete Logarithm (DL) Assumption.*

Let λ be a security parameter and $G =< g >$ denotes a cyclic group of prime order p. DL problem [33] is that, given a tuple $(g, g^a) \in G$ to output $a \in Z_p$, where Z_p is the set of non-negative integers smaller than p. DL assumption holds if for any polynomial-time algorithm $\mathcal{A}$, the following advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DL}}$ is negligible in λ,

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DL}}(\lambda) = Pr\Big[\mathcal{A}(g, g^a) \rightarrow a\Big].$$

2) *Decisional Diffie-Hellmam (DDH) Assumption*

Let λ be a security parameter and $G =< g >$ denotes a cycle group of prime order p. DDH problem [33] states that given a tuple $(g, g^a, g^b, g^{(1-x)ab+xc}) \in G$ and output $x \in \{0, 1\}$. DDH assumption holds if for any polynomial-time algorithm $\mathcal{C}$, the following advantage $\mathbf{Adv}_{\mathcal{C}}^{\mathbf{DDH}}(\lambda)$ is negligible in λ.

$$\mathbf{Adv}_{\mathcal{C}}^{\mathbf{DDH}}(\lambda) = \Big|Pr\Big[\mathcal{C}(g, g^a, g^b, g^{ab}) = 1\Big] - Pr\Big[\mathcal{C}(g, g^a, g^b, g^c) = 1\Big]\Big|.$$

3.2 Distributed ElGamal Encryption

ElGamal encryption [30] is semantically secure under the DDH assumption. Another merit of ElGamal encryption is its inherent homomorphism. The ciphertexts of m_0, m_1 can be easily aggregated to obtain the ciphertext of $m_0 m_1$. A distributed ElGamal cryptosystem [31] is a generalization of ElGamal encryption, which contains the following algorithms.

Setup. Suppose there are n users in the system, and the key pairs of the ith user are $(x_i, y_i = g^{x_i})$. Each user publishes his public key, and the common public key can be generated in a distributed manner [32] as $y = \prod_{i=1}^{n} y_i$.

Enc. To encrypt a message m, randomly choose r and compute a ciphertext (c_1, c_2) of m as $(g^r, y^r \cdot g^m)$.

Dec. Each user computes and broadcasts the partial decryption key $c_1^{x_i}$. Then the decryption can be done by computing

$$g^m = c_2 / \prod_{i=1}^{n} {c_1}^{x_i} = c_2 / {c_1}^{x_1 + \cdots + x_n}.$$

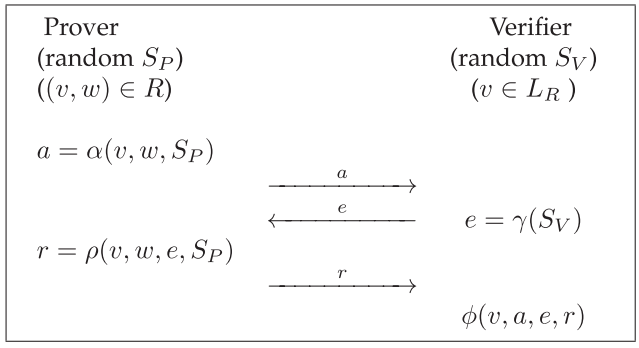


Fig. 1. Σ-protocol.

3.3 Commitment

A commitment scheme allows a user to commit to a selected statement, which is hidden to others during the *Commit* phase, but can be revealed by the user in the *Open* phase. A commitment scheme owns the following two properties [33]:

- *Binding*. The committer cannot change the statement after he commits to the statement.
- *Hiding*. The receiver knows nothing about the committed statement before the committer opens the commitment.

3.4 Zero-Knowledge Proof of Knowledge and Σ-Protocol

Let $R = (x, w)$ be a binary relation, where x is the common input and w is a witness. A zero-knowledge proof of knowledge is a protocol in which a prover $\mathcal{P}$ proves to a verifier $\mathcal{V}$ that it knows a witness w for which $(x, w) \in R$ without revealing anything.

A Σ-Protocol is a way to design an efficient zero-knowledge proof. A protocol is a Σ-Protocol for relation R if it has 3-move as shown in Fig. 1. 1) $\mathcal{P}$ sends a commitment a to $\mathcal{V}$. 2) $\mathcal{V}$ sends a random t-bit challenge e to $\mathcal{P}$. 3) $\mathcal{P}$ sends a response r, and $\mathcal{V}$ decides to accept or reject based on the verification algorithm. A Σ-protocol has the following properties.

- Completeness. If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on input x and w, where $(x, w) \in R$, the verifier always accepts the prover's proof.
- Special soundness. For any x and any accepting conversations on x with the same commitment a and

8. https://followmyvote.com/
9. https://tivi.io/
10. http://www.smartmatic.com/fileadmin/user_upload/Whitepaper_Online_Voting_Challenge_Considerations_TIVI.pdf
11. https://github.com/stonecoldpat/anonymousvoting
12. https://ethereumfoundation.org/devcon3/sessions/the-open-vote-network-decentralised-internet-voting-as-a-smart-contract/
13. https://citizendata.network/netvote/.

different challenges (a, e, r) and (a, e', r'), where $e \neq e'$, one can efficiently extract w such that $(x, w) \in R$.

- Honest verifier zero-knowledge (HVZK). There is a polynomial-time simulator, which on input x and a challenge e outputs an accepting conversation with the form (a, e, r), which has the same probability distribution as conversations between the honest $\mathcal{P}$ and $\mathcal{V}$ on input x.

The special soundness property implies that the error probability of this proof system is always 2^{-t}.

A Σ-protocol is efficient to prove AND, OR and arbitrary combinations of AND/OR statements. More details can be found in [29], [35], [36], [37].

3.5 Blockchain

Blockchain [1] was proposed in 2008 as the backbone technology of cryptocurrencies to achieve decentralization. Blockchain is a public ledger that records all the modifications in the system as transactions. The logged transactions cannot be removed and can be accessed by all legitimate users in the system. In a nutshell, the blockchain system works as follows. Each user gets a public-secret key pair in the system, in which the public keys are the users' identities. Users in blockchain system can conduct transactions, which include the details of the modification to the system, some necessary information (timestamp and etc.) and a signature. The validation of the transactions can be checked with the corresponding public keys. Miners choose some transactions from the mining pool and generate a block. The block can be logged into the blockchain when the miners solve some predefined hard problems such as proof-of-work. The blocks are broadcast to all the users in the system once it is onchain. Blockchains can be classified into three categories, namely public blockchains, consortium blockchains and private blockchains. In public blockchains, users can freely join or leave the system, such as the bitcoin blockchain [1] and Ethereum [2]. In consortium blockchains and private blockchains, users need approval to enroll in the system.

4 System and Security Model

In this section, we describe the system model of the blockchain-based self-tallying voting system for decentralized IoT and list the necessary security requirements and the security model of a self-tallying voting protocol.

4.1 System Model

The framework of a blockchain-based self-tallying voting protocol for a decentralized IoT system is shown in Fig. 2. There are three roles in the system, namely smart devices, a gateway and a blockchain. The IoT system is equipped with a number of smart devices, which are regarded as voting devices. A blockchain is leveraged to achieve a P2P overlay network and can also fulfill device management [5] and a bulletin board. Each device needs to register when they first enroll in the system and cast ballots through the gateway to the blockahin. After collecting the ballots from the blockchain, the results can be obtained immediately to make decisions for the whole IoT system. Note that, the blockchain leveraged in the model can be a private blockchain or a consortium blockchain

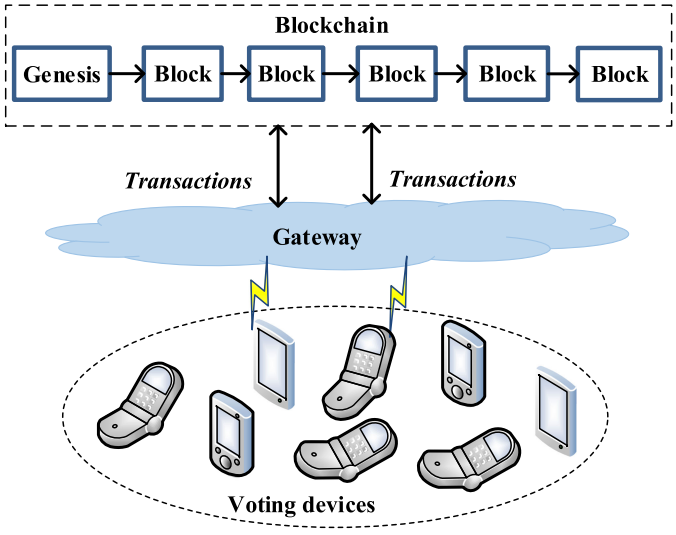


Fig. 2. The framework of the blockchain-based self-tallying voting system.

(according to different voting scenarios) rather than a public blockchain, which enjoys efficient consensus in practice, like practical byzantine fault tolerance (PBFT) [46]. We can also designate a block generator to generate new blocks if a private blockchain is suitable for the application. We also note that the security of blockchain in our voting system matters a lot, which is the foundation of the security of the whole system. Blockchain also plays an important role in the security guarantee of the voting protocol.

4.2 System Components

Suppose there are n voting devices in the system denoted as voter $\overline{V_i}$, where i is the counting variable from 1 to n. We denote x_i a variable x for the voter $\overline{V_i}$ and $\{x_i\}_{i\in n}$ the set of all the variables for each voter $\overline{V_i}$. A blockchain-based self-tallying voting system in decentralized IoT consists of the following algorithms.

$Setup(k, n) \rightarrow (sk_i, pk_i)$. This is a probabilistic algorithm that takes a security parameter k and the number of voters n as input and outputs the private and public key pair (sk_i, pk_i) for each voter $\overline{V_i}$.

$Commit(v_i, \{pk_j\}_{(j\neq i, j\in n)}) \rightarrow (C_i)$. This algorithm is run by each voter $\overline{V_i}$. On input a vote v_i and all the other voters $\overline{V_j}$'s public key $\{pk_j\}_{(j\neq i)}$, it outputs a commitment C_i and a corresponding zero-knowledge proof. C_i and the proof will be published on the blockchain.

$Vote(v_i, sk_i, \{pk_j\}_{(j\neq i, j\in n)}) \rightarrow (V_i)$. This algorithm is run by each voter $\overline{V_i}$. On input a vote v_i, the private key sk_i, and the other voter $\overline{V_j}$'s public key $\{pk_j\}_{(j\neq i, j\in n)}$, it outputs a ballot V_i and a zero-knowledge proof to prove the ballot is in the right form (a.k.a. follow the protocol), and publishes V_i and the proof on the blockchain.

$Tally(\{V_i\}_{(i\in n)}) \rightarrow (Result)$. This is a deterministic algorithm that takes all the ballots $\{V_i\}_{(i\in n)}$ as input, and outputs the election result $Result$.

$Recover(\{sk_j\}_{(j\neq i, j\in n)}, \{C_i\}_{(i\in n)}) \rightarrow (v_i)$. This algorithm is to recover the abortive voter's vote. On input the abortive voter's commitment C_i and all the other voters' private key $\{sk_j\}_{(j\neq i, j\in n)}$, it outputs the abortive voter's vote v_i.

4.3 Attack Model

We consider two types of adversaries in our system: the passive adversaries and the active adversaries. The passive adversaries would not actively involve in the voting process, but only eavesdrop from the communication channel and/or the blockchain, trying to get the knowledge of the ballots. Active adversaries could actively hinder or manipulate the voting, and can abort before the voting finishes or collude with other voters to get more information about the ballots.

4.4 Security Requirements

A self-tallying protocol is supposed to satisfy the following four security requirements against the attack model defined above, in which the first one is to resist passive adversaries and the other three are against active adversaries.

- *Maximal ballot secrecy*. A partial tally of the ballots can be accessed only by collusion of all remaining voters.
- *Self-tallying*. After all the voters cast their ballots, anyone is able to compute the voting results with all the ballots.
- *Fairness*. Fairness means that nobody has the priority to get a partial tally ahead of schedule. Self-tallying protocols always suffer from fairness issues, including abortive issues and adaptive issues. Abortive issues indicate that some of the users refuse to reveal their votes and abort before casting their ballots, then the final results won't be revealed. Adaptive issues state that the last voter has the priority to know the final results in advance, which may affect his choice or make him abort, causing an abortive issue.
- *Dispute-freeness*. This property states that anyone can check whether the voters follow the protocol or not. This is an extension of universal verifiability.

4.5 Security Model

In this section, we formalize the security model for *maximal ballot secrecy*.

Suppose there are maximal $n-2$ corrupted voters in the *maximal ballot secrecy* game, who are fully controlled by the adversary, since $n-1$ collusive voters can easily get the information of the last voter according to the final result in the game. The adversary can make queries to the commitments as well as the corrupted users' ballots, and also get the final result of the election. And later in the challenge phase, given two ballots from different votes $\{0,1\}$ for the two uncorrupted voters, the adversary needs to tell which of the two ballots is from the vote 1. The detailed security model between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is as follows.

Maximal ballot secrecy (MBS). We say a self-tallying voting scheme is MBS-secure, if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against a challenger $\mathcal{C}$ in the following game.

Initial. There are n voters in the game. $\mathcal{A}$ declares two target voters $\overline{V_s}, \overline{V_t}$ to be challenged upon. The other voters are regarded as corrupted users, whose votes are all controlled by $\mathcal{A}$. $\mathcal{C}$ randomly chooses $\overline{V_b}$ from $\{\overline{V_s}, \overline{V_t}\}$ and set the vote of $\overline{V_b}$ as 1, and the other voter's vote as 0.

Setup. $\mathcal{C}$ generates the private and public key pairs for each voter. Then $\mathcal{C}$ forwards all the public keys and the corrupted users' private keys to $\mathcal{A}$.

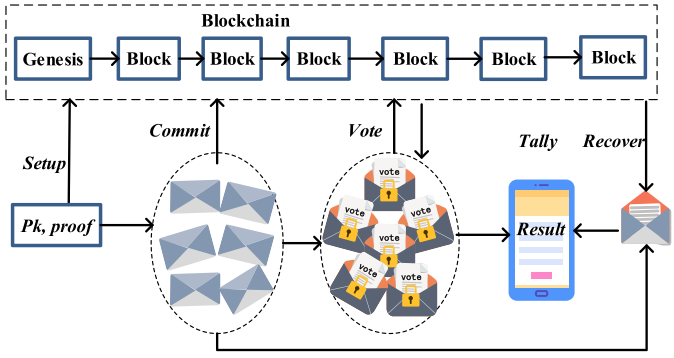


Fig. 3. The workflow of the blockchain-based self-tallying voting system.

Queries. $\mathcal{A}$ can choose any ballots for the corrupted users and make some queries including the Commit queries and the Vote queries corresponding to the chosen ballots.

- Commit queries. $\mathcal{A}$ can query the commitment for a vote. Then $\mathcal{C}$ generates the commitment and records the ballot and the commitment in the list $\mathcal{L}_{\mathcal{C}}$.
- Vote queries. $\mathcal{A}$ can make queries on the votes generated by any user other than $\overline{V_s}, \overline{V_t}$.

Challenge. $\mathcal{C}$ outputs two challenge ballots on behalf of the uncorrupted voters $\overline{V_s}$ and $\overline{V_t}$ chosen in the *Initial* phase.

Tally. $\mathcal{A}$ computes the final result of the election according to the collected ballots.

Guess. $\mathcal{A}$ outputs a guess $guess$ to determine which one between $\overline{V_s}$ and $\overline{V_t}$ has cast the ballot of 1.

In the above model, the reason we set two challenge ballots rather than one is to prevent the adversary deducing the challenged vote from his known information. Specifically, the adversary can control the ballots of the corrupted voters and obtain the election result after collecting the challenge vote, if there is only a single challenge vote, the adversary can have a non-negligible advantage in the guessing game. After collecting the votes together, the adversary can do the tallying by itself to know the election result. To see why we set the different ballots for the two challenge votes, let's suppose the following situation. If we set the challenge vote with the same ballot from $\{0,1\}$, and all the corrupted voters controlled by the adversary vote the same ballot, then the adversary can get the knowledge about the challenge vote easily after knowing the results, in which the advantage ϵ is non-negligible.

Definition 1. *The voting scheme is MBS secure if for any polynomial-time adversary,*

$$|Pr[guess = \overline{V_b}] - 1/2| \leq \epsilon,$$

where ϵ is negligible.

5 Construction

In this section, we present a concrete construction of the self-tallying voting system assisted by blockchain. As shown in Fig. 3, the system contains three phases, Pre-vote phase, which includes *Setup* and *Commit* algorithms, Vote phase, which includes *Vote* algorithm, and After-vote phase, which includes *Tally* and *Recover* algorithms. In Pre-vote phase, the system is initialized and the voters register to obtain their

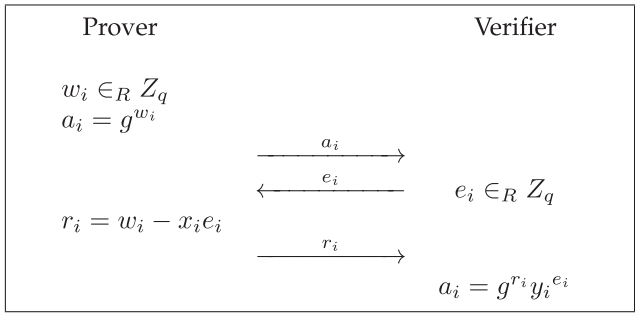


Fig. 4. Zero-knowledge proof for setup.

private-public key pairs. Voters put their public keys together with the zero-knowledge proofs for the corresponding private keys on the blockchain. *Commit* is to ensure fairness. If voters skip the commit part and cast their ballots directly, the last voter has the priority to access the final result ahead of schedule. In this phase, other voters cannot see the vote but only the commitment of the vote, thus, the voters need zero-knowledge proofs to prove the committed vote is in the right form (follow the protocol). Later, if the last voter refuses to vote, other voters can recover the ballot according to commitment and get the result. In Vote phase, the voters cast their encrypted ballots. In After-vote phase, by collecting all the ballots from the blockchain, the final result can be obtained publicly. *Recover* is optional which is called when the last voter does not follow the rules to cast his/her ballot. The ballot of the last voter can be recovered with the corresponding commitment and the assistance of all the other voters.

5.1 Dealing With Abortive Issues

Basic Idea. The existing approaches to deal with the abortive issue are adding a recovery phase, in which the abortive users are excluded by removing their ballots, and tallying the ballots from the remaining voters. However, an abort may be caused by some user who knows the unwanted result and is against revealing the result. So, simply removing the votes might lead to a different voting result. Thus, we modify the recovery phase in [22]. In our modification, if the last voter quits after making a commitment, then his/her ballot can be revealed according to the corresponding commitment with the cooperation of all the other voters. The detailed construction is as follows.

Setup$(k, n) \rightarrow (x_i, y_i)$. On input a security parameter k, and the number of voters n, it initializes the system by choosing two large prime p and q, where q is the divisor of $p-1$. Z_p^* is a cyclic group modular p and Z_q is the subgroup of order q. Each voter $\overline{V_i}$ chooses a random private key $x_i \in Z_q^*$, and computes the public key g^{x_i}. Then $\overline{V_i}$ generates a zero-knowledge proof as $\text{ZKPoK}_1\{(x_i) : y_i = g^{x_i}\}$(cf. Fig. 4). The public key and the corresponding zero-knowledge proof are published to blockchain.

Commit$(v_i, \{y_j\}_{(j\neq i, j\in n)}) \rightarrow (C_i)$. Before casting a ballot, each voter $\overline{V_i}$ collects the other voters' public key $y_{j(j\neq i)}$. To generate a commitment to the vote, $\overline{V_i}$ chooses a random ρ_i and publishes $\beta_i = g^{\rho_i}$. $\overline{V_i}$ makes the commitment $C_i = g^{v_i} Y_i^{\rho_i}$ to ensure fairness, where v_i is the vote from {0,1} and $Y_i = \prod_{j=1, j\neq i}^{n} y_j$. The voters also need to generate a zero-knowledge proof to prove that the commitment is in the right form (cf. Fig. 5) as

$$ZKPoK_2\{(\rho_i) : (C_i = {Y_i}^{\rho_i} \vee C_i = g \cdot {Y_i}^{\rho_i}) \wedge \beta_i = g^{\rho_i}\}.$$

And then the commitment and zero-knowledge proof are put on the blockchain.

Vote$(v_i, x_i, \{y_j\}_{(j\neq i, j\in n)}) \rightarrow (V_i)$. To ensure the secrecy of the vote, all voters encrypt their votes as $V_i = {h_i}^{x_i} g^{v_i}$, where $h_i = \prod_{j=1}^{i-1} y_j / \prod_{j=i+1}^{n} y_j$. A zero-knowledge proof is generated to prove that the vote v_i is the same as the one committed in the commitment. The statement (cf. Fig. 6) is as follows.

$$\begin{aligned} ZKPoK_3\{(x_i, \rho_i) : &(C_i = {Y_i}^{\rho_i} \wedge V_i = {h_i}^{x_i} \wedge y_i = g^{x_i} \wedge \beta_i = g^{\rho_i}) \\ &\vee (C_i = g \cdot {Y_i}^{\rho_i} \wedge V_i = g \cdot {h_i}^{x_i} \wedge y_i = g^{x_i} \wedge \beta_i = g^{\rho_i})\}. \end{aligned}$$

Then publish the ballot on the blockchain.

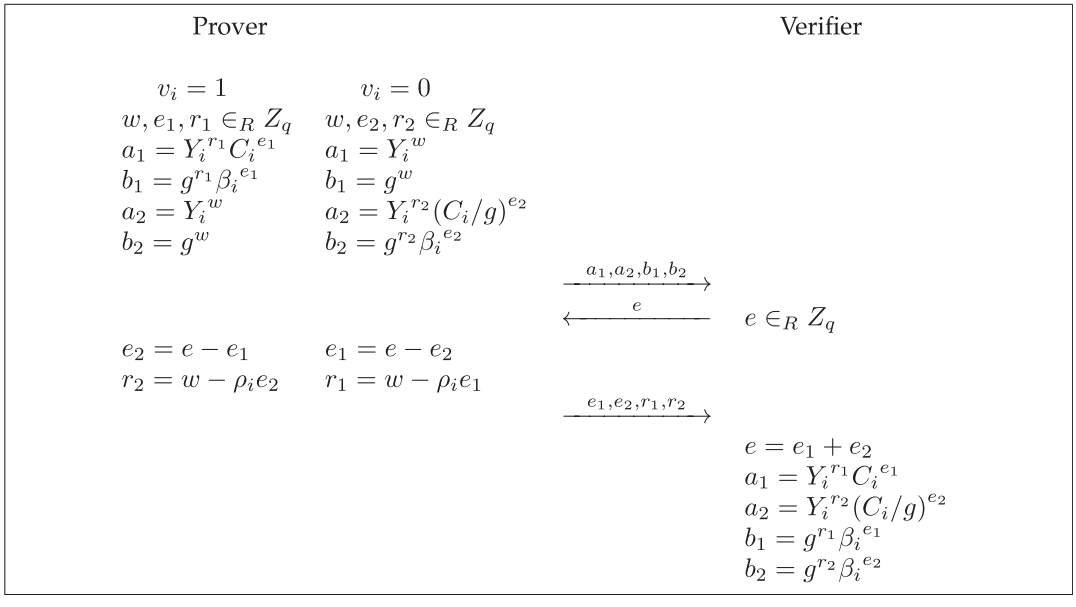


Fig. 5. Zero-knowledge proof of knowledge for Commit

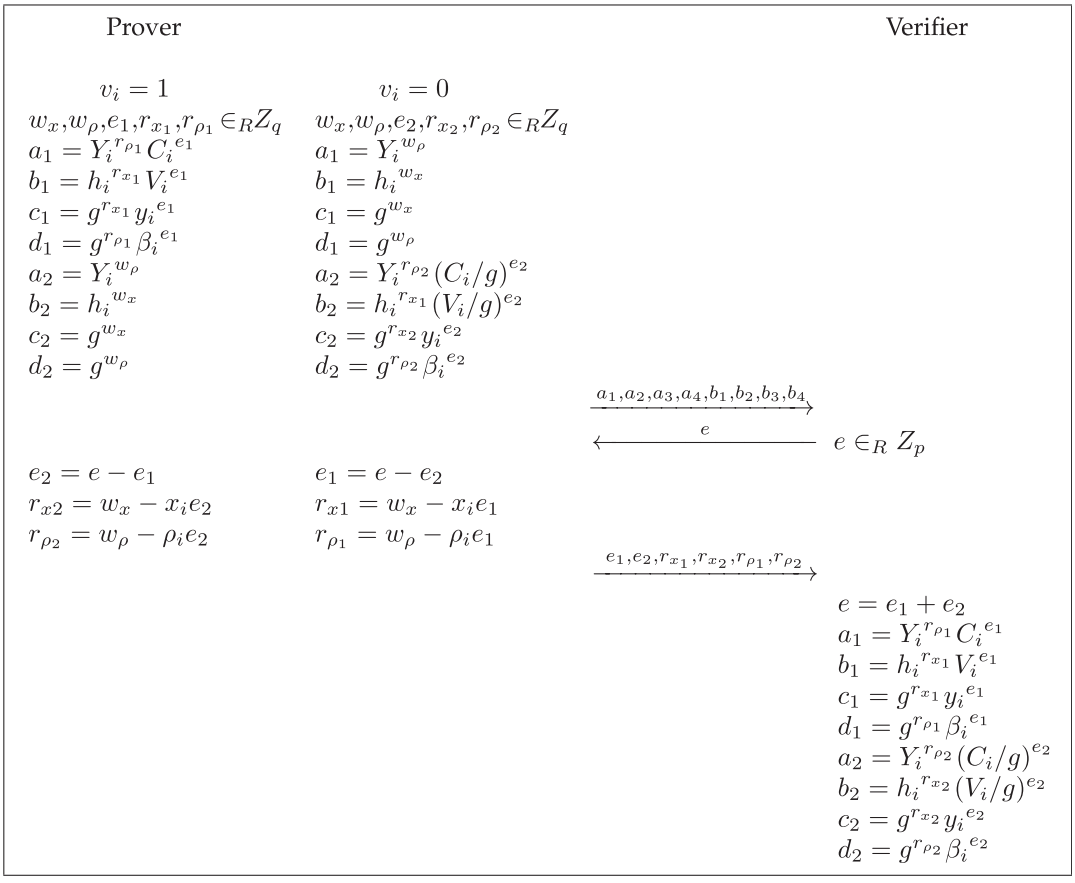


Fig. 6. Zero-knowledge proof of knowledge for Vote.

Tally$(\{V_i\}_{(i\in n)}) \rightarrow (Result)$. To tally the votes, one collects all the ballots and computes $\prod_{i=1}^{n} V_i = \prod_{i=1}^{n} h_i{}^{x_i} g^{v_i} = g^{\sum_{i=1}^{n} v_i}$. As $\sum_{i=1}^{n} v_i$ is within a small set, the result $Result$ can be easily obtained in a brute-force manner.

Recover$(\{x_j\}_{(j\neq i, j\in n)}, \{C_i\}_{i\in n}) \rightarrow (v_i)$. If the last voter $\overline{V_i}$ does not cast his ballot in Vote phase, then each of the remaining voters $V_j (1 \leq j \leq n, j \neq i)$ publish a recover factor for $\overline{V_i}$ as $R_{ij} = y_j{}^{\rho_i} = \beta_i{}^{x_j}$ together with a zero-knowledge proof to prove that it is in the right form (cf. Fig. 4). The value of g^{v_i} can be computed as $g^{v_i} = C_i / \prod_{j=1, j\neq i}^{n} R_{ij} = C_i / \prod_{j=1, j\neq i}^{n} y_j{}^{\rho_i}$. Then the value of v_i is easy to get as there are only two candidates.

To compute the final result of the election, each remaining voter $\overline{V_j}$ publishes $\hat{h_j}{}^{x_j}$, where $\hat{h_j} = \prod_{k=j+1, k\neq i}^{n} y_j / \prod_{k=1, k\neq i^{j-1}} y_j$, and a ZKPoK to prove the knowledge of x_j as in Fig. 4. Now everyone can compute the result of the remaining voters as $g^{\sum_{j\neq i} v_j} = \prod_{j\neq i} \hat{h_j}{}^{x_j} V_j$. So the final result of this election is $\sum_{j\neq i} v_j + v_i$.

We note that the proofs in Figs. 5 and 6 are three-move interactive protocols with the techniques in [35], which can be transformed into non-interactive protocols following Fiat-Shamir's heuristics [26] by setting e to be a hash value of a secure hash function.

5.2 Dealing With Adaptive Issues

The adaptive issues seem inevitable in self-tallying protocols from its definition because the last voter holding the ballot has the priority to access the final results ahead of the other voters. We suggest using time-locked primitives [38], [39] to deal with the adaptive issues in voting systems. Time-lock encryption allows users to get the results only after a certain time [27], [28]. Once the deadline is passed, the decryption can be performed immediately. It is stated in [27] that time-locked encryption can be achieved by using witness encryption with blockchain as the computational reference clock. We borrow this idea in our protocol by encrypting the vote with witness encryption and the witness can be produced by blockchain after a certain time. And the blockchain can also act as the computational reference clock to measure the "certain" time, say after generating certain blocks. Then the votes can be decrypted once the deadline is passed and thus all the voters and observers can do the tallying to obtain the voting result simultaneously.

6 Security Proof

In this section, we show that the proposed protocol satisfies all the security requirements presented in Section 4.

First, we show that the zero-knowledge proofs of knowledge in Figs. 4, 5, and 6 satisfy completeness, special soundness [35], [36] and honest verifier zero-knowledge (HVZK). We show the detailed proof of Fig. 5 as an example and omit the proofs in other figures, since the proofs are quite similar.

Theorem 1. *The zero-knowledge proof in Fig. 5 satisfies completeness, special soundness and honest verifier zero-knowledge.*

Proof. We omit the proof for completeness as it's straightforward to verify. □

The witness for the statement in $ZKPoK_2$ is ρ_i. To prove special soundness, the goal is to extract a witness from the three-move interaction with two accepting conversations in polynomial time. Given the two accepting conversations with the same values in the first round, different random numbers in the second round and different responses in the third round as $(a_1, a_2, b_1, b_2, e, e_1, e_2, r_1, r_2)$ and $(a_1, a_2, b_1, b_2, e, e'_1, e'_2, r'_1, r'_2)$, it can be checked easily that one of the following holds $\rho'_i = (r_1 - r'_1)/(e_1 - e'_1)$ or $\rho'_i = (r_2 - r'_2)/(e_2 - e'_2)$.

To prove HVZK, assume there exists a simulator $\mathcal{S}$, who is given a random e. It randomly chooses r_1, r_2, e_1, e_2, where $e = e_1 + e_2$, and computes the conversation as $(Y_i{}^{r_1} C_i{}^{e_1}, Y_i{}^{r_2} C_i / g^{e_2}, g^{r_1} \beta_i{}^{e_1}, g^{r_2} \beta_i{}^{e_2}, e, e_1, e_2, r_1, r_2)$, which is an accepting conversation. It is indistinguishable from the one generated by the honest prover.

Next, we prove the proposed scheme is MBS secure if ZKPoK is zero-knowledge and the DDH assumption holds.

Theorem 2. *If there exists an adversary that can win the guessing game in the MBS security model with a non-negligible advantage, then we can build an algorithm $\mathcal{B}$ that can break the zero-knowledge of the ZKPoK and the DDH problem.*

Proof. Suppose there are n voters $\overline{V_1}, \ldots, \overline{V_n}$ in the game. The challenger $\mathcal{C}$ can interact with the adversary $\mathcal{A}$. We list a sequence of games [40] to prove Theorem 2. We denote $\Pr[\mathsf{Win}_i]$ as the winning probability of an adversary (output a correct guess) in Game_i. □

Game 0: This is the original Game defined in Section 4.5. $\mathcal{A}$ chooses two target voters $\overline{V_s}, \overline{V_t}$ to challenge upon and forwards them to $\mathcal{C}$. $\mathcal{C}$ tosses a coin to decide that one of the voters from $\{\overline{V_s}, \overline{V_t}\}$ votes 1 and the other one votes 0. The reason that we do in this way is to let $\mathcal{A}$ knows nothing even from the tally result. The one who votes 1 is denoted by $\overline{V^*}$. The challenges are denoted as $\{C^*_s, \pi^*_s, V^*_s\}$ and $\{C^*_t, \pi^*_t, V^*_t\}$, where C^*_k is the commitment of the vote, π^*_k represents all the ZKPoK in the scheme, V^*_k is the ballot, $k \in \{s, t\}$. The adversary outputs a guess $guess$, then from the definition of the MBS game, we have

$$\Pr[\mathsf{Win}_0] = \Pr[guess = \overline{V^*}].$$

Game 1. Game 1 is the same as Game 0 with one difference. $\mathcal{C}$ runs a simulator $\mathcal{S}$ as in Theorem 1, and replaces all the zero-knowledge proofs (π^*_s, π^*_t) with the simulated proofs (π, π') without using the real witness. The setting is indistinguishable from $\mathcal{A}$'s view. If $\mathcal{A}$ can distinguish between the two settings in *Game 0* and *Game 1* with a non-negligible advantage, then we can use the adversary to construct an algorithm $\mathcal{B}$ to violate Zero-Knowledge of ZKPoK. Thus, the adversary's winning probability in *Game 1* satisfies the following equation.

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq \epsilon_{ZK}.$$

Game 2. Game 2 is the same as Game 1 with one difference. $\mathcal{C}$ replaces the commitment C^*_s with a random number C_s. The two settings are indistinguishable from $\mathcal{A}$'s view. Specifically, $\mathcal{C}$ generates private and public key pairs for the voters other than $\{\overline{V_s}, \overline{V_t}\}$. Then set the public key for $\overline{V_t}$ as g^a, β_s as g^b, $R \in \{g^{ab}, g^r\}$, where r is a random number. $\mathcal{C}$ sets $C'_s = g^{v_s} R \cdot (g^b)^{\sum_{i=1, i\neq s,t}^{n} x_i}$ Clearly, if there is a difference in the adversary's winning probability between *Game 1* and *Game 2*, we can use the adversary to construct an algorithm $\mathcal{B}$ to violate DDH problem. Thus, the adversary's winning probability in *Game 2* satisfies the following equation.

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \epsilon_{DDH}.$$

Game 3. Game 3 is the same as Game 2 with one difference. $\mathcal{C}$ replaces the commitment C^*_t with some random number C'_t. Following the same analysis as in the previous game, $\mathcal{C}$ sets the public key of $\overline{V_s}$ as g^a, β_t as g^b, $R \in \{g^{ab}, g^r\}$, where r is a random number. $\mathcal{C}$ sets $C'_t = g^{v_t} R \cdot (g^b)^{\sum_{i=1, i\neq s,t}^{n} x_i}$. If there is a difference in the adversary's winning probability between *Game 2* and *Game 3*, we can use the adversary to construct an algorithm $\mathcal{B}$ that violates the DDH problem. Thus, the adversary's winning probability in *Game 3* satisfies the following equation.

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq \epsilon_{DDH}.$$

Game 4. Game 4 is the same as Game 3 with one difference. $\mathcal{C}$ changes the values of V^*_s, V^*_t with two random elements V'_s and V'_t satisfying a certain relation. The change is indistinguishable from $\mathcal{A}$'s view under the DDH assumption. Wlog, we assume $s < t$. Given the DDH instance $(A = g^a, B = g^b, C)$ where $C \in \{g^{ab}, g^r\}$, $\mathcal{C}$ sets the public key of $\overline{V_t}$ and $\overline{V_s}$ as $A = g^a$ and $B = g^b$ respectively. $\mathcal{C}$ computes V'_s and V'_t as

$$V'_s = g^{v_s} A'/C, \qquad V'_t = g^{v_t} B' C$$

where

$$A' = A^{\sum_{j=1}^{s-1} x_j - \sum_{j=s+1}^{t-1} x_j - \sum_{j=t+1}^{n} x_j},$$
$$B' = B^{\sum_{j=1}^{s-1} x_j + \sum_{j=s+1}^{t-1} x_j - \sum_{j=t+1}^{n} x_j}.$$

Thus, V'_s and V'_t are two random elements satisfying $V'_s = gA'B'/V'_t$. Clearly, if there is a non-negligible difference in the adversary's winning probability between *Game 3* and *Game 4*, we can use the adversary to construct an algorithm $\mathcal{B}$ to solve DDH problem. Thus, the adversary's winning probability in *Game 4* satisfies the following equation.

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq \epsilon_{DDH}.$$

Wrapping Up. The winning probability for $\mathcal{A}$ in *Game 4* to output a correct *guess* is $1/2$ because in this game the challenges contain only random numbers, which are independent of the votes v_s, v_t. Therefore, we can conclude that there is only a negligible difference in winning probability for an adversary between *Game 0* and *Game 4*, if all the ZKPoKs in the scheme are zero-knowledge and DDH assumption holds. So the probability that $\mathcal{A}$ wins the *MBS* game is $\frac{1}{2} + \epsilon$, where $\epsilon = \epsilon_{ZK} + 3\epsilon_{DDH}$.

Now, we show that the scheme satisfies fairness, self tallying and dispute freeness as well.

Fairness. Suppose voter $\overline{V_i}$ votes for v_i in the *Commit* phase and refuses to provide the vote in the *Vote* phase.

TABLE 1
Comparisons Among the Existing Self-Tallying E-Voting

	self-tallying	privacy	fairness	robustness	rounds
KY02 [18]	✓	✓	✓	✓	c
Groth [19]	✓	✓	✓	×	n+1
Hao [20]	✓	✓	×	×	c
Khader [21]	✓	✓	×	✓	c
Our proposal	✓	✓	✓	✓	c

- *c represents constant and n is the number of voters.*

TABLE 2
Computational Complexity Analysis

Protocols	*exp*	ZKP for *exp*	ZKP for AND	ZKP for OR
[18]	2n+2	n+1	n	1
[19]	4	2	1	1
[20]	2	1	0	1
Ours	2	1	4	2

TABLE 3
Communication Overhead Analysis

Setup	Commit	Vote	Tally	Recover
$3\|Z_p\|+\|Z_q\|$	$6\|Z_p\|+4\|Z_q\|$	$9\|Z_p\|+7\|Z_q\|$	$(n-1)\|Z_p\|$	$(n+3)\|Z_p\|+\|Z_q\|$

Due to the Soundness of ZKPoK, we can guarantee that v_i is decryptable by other voters in the *Recover* phase.

Self-Tallying. The zero-knowledge proof of knowledge in each algorithm in the proposed protocol forces the voters to perform honestly according to the protocol. After all the voters cast their ballots, the self-tallying property is easy to verify. Since $\prod_{i=1}^{n} {h_i}^{x_i} = 0$ in the *Tally* algorithm, the self-tallying property is achieved.

Dispute Freeness. To dispute freeness, again, the zero-knowledge proof of knowledge in each algorithm of the proposed protocol ensures that the commitments and ballots are generated in the right form and can be publicly verified.

7 Performance Evaluation

In this section, we first analyze the properties, the computational complexity and communication overhead of the proposed protocol, and then report the implementation results of each algorithm. We also test the gas cost of each algorithm on a private blockchain.

7.1 Protocol Analysis

A comparison among the existing self-tallying protocols is provided in Table 1. We focus on several main properties that a self-tallying voting protocol should have, including privacy, fairness, robustness. The number of rounds in the protocol is also considered in the table. From Table 1, we can see that our protocol satisfies all these properties. [18] also has a good performance in this table, however, the efficiency is not good enough.

The computation complexity analysis is provided in Table 2. The parameters in Table 2 are elaborated as follows. Assume there are n voting machines and 1 of them aborts in the Vote phase. We only count the expensive operations and ignore the cheap ones. *exp* denotes the exponentiation operation. ZKP for *exp* denotes the zero-knowledge proof to prove the knowledge of an exponent. ZKP for AND denotes the zero-knowledge proof to prove several statements about discrete logarithms are true simultaneously. ZKP for OR denotes the zero-knowledge proof to prove the 1-out-of-2 statement about discrete logarithm is true. We can see from the table that our proposal is more efficient than the first two protocols and also comparable to the third one. As in [20], if a voter refuses to cast his ballot, the voting would abort and restart the whole protocol. Our protocol improves the conditions by adding the Commit and Recover, which costs some additional computation but is acceptable.

The communication overhead for each algorithm is analyzed in Table 3, in which $|Z_p|$ and $|Z_q|$ represent the length of the element in the group Z_p and Z_q, respectively.

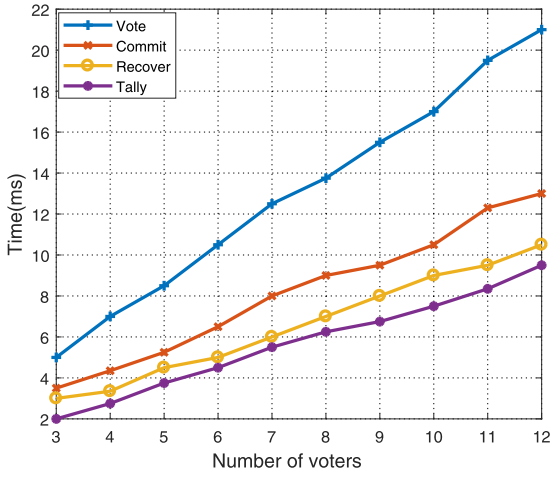


Fig. 7. Simulation on laptop.

7.2 Implementation Results

We also implement our proposal to test the time consumption of each algorithm in a variety of test environments. In our experiments, we first implement the protocols on a laptop (Fig. 7). For a better simulation of IoT devices, we then run the protocols on a mobile phone (Fig. 8), which has constrained resources. Besides, Raspberry Pi is regarded as the

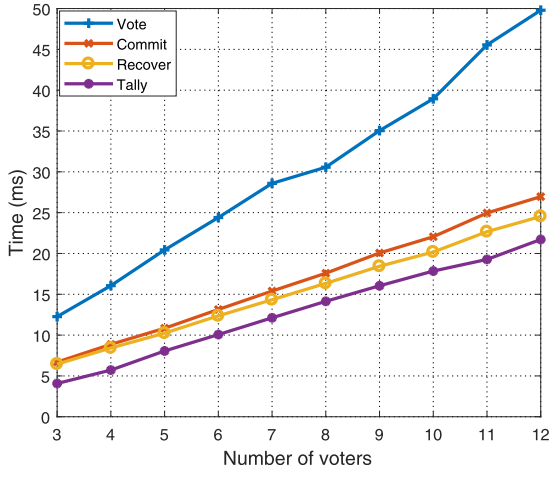


Fig. 8. Simulation on android device.

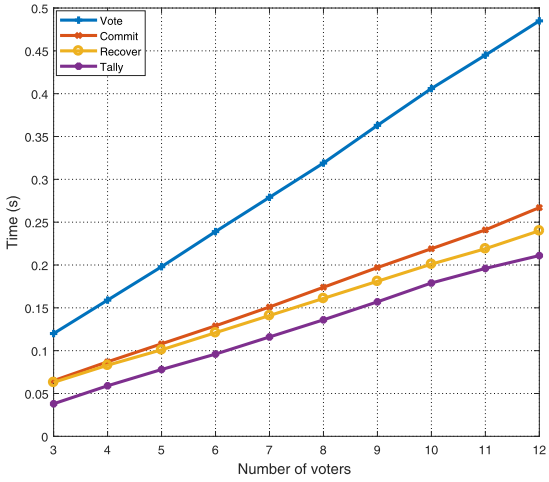


Fig. 9. Simulation on raspberry Pi.

super platform to build IoT projects. Thus, we also evaluate the simulations on a Raspberry Pi 3 Model B+ (Fig. 9).

Environment. The running environment of the laptop is with Win 8 64-bit operating system and Intel Core (TM) i5-4300 @2.49 GHz CPU with an 8 GB RAM. And the configuration of the phone is an Android 7.1.1 operating system with Qualcomm MSSM8998 @2.45 CPU (Octa-core) and a 6 GB RAM. The Raspberry Pi is equipped with Broadcom BCM2837B0, 1.4 GHz 64-bit quad-core ARM Cortex-A53 CPU and 1 GB LPDDR2 SDRAM. The operating system for the Raspberry Pi is Raspbian with kernel v4.14, which is the recommended operating system based on Debian. The projects are written in C++ language with Miracl library [14] under Visual Studio 2010 for the laptop and Android Studio compiler for the mobile phone, respectively. The project in Raspberry Pi is with the help of GMP library.[15] We test the efficiency of each algorithm with the increasing number of voters. The implementation results are illustrated as follows.

Implementation Results. In our experiment, we set the number of voters from 3 to 12 to test the efficiency of each algorithm. As we can see from the three figures (Figs. 7, 8, and 9), the trend of algorithms is almost the same on each platform, but the magnitude is different, which shows different processing capability of the three kinds of devices. The time consumption of all the algorithms is linear with the number of the voters, as the more voters in the system, the more computations are required for each algorithm. In the four algorithms, the most expensive one is Vote, as $ZKPoK_3$ is dominant in Vote and it is the most complicated one among all the zero-knowledge proofs, which costs 21.03 ms, 49.794 ms and 0.48 s for 12 voters on the laptop, the Android phone and the Raspberry Pi, respectively. The most efficient algorithm is Tally, which is consistent with our theoretical analysis, as no zero-knowledge proof is needed and the equation to tally the votes is the product of the voters' ballot, which is linear with the number of voters. The running time of Tally for 12 voters is 4.076 ms, 21.714 ms, and 0.21 s on these three platforms. For the other two algorithms, Commit and Recover, the expensive zero-knowledge proofs are needed in these two algorithms, but are not as complicated as the one in Vote. The time costs for Commit on these three platforms for 12 voters are 12.264 ms, 49.794 ms and 0.27 s respectively. When it comes to Recover, the time consumption is 10.2 ms, 24.8 ms and 0.243 s, respectively on the three platforms.

14. https://certivox.org/display/EXT/MIRACL.
15. www.gmp.org.

TABLE 4
Gas Cost for Each Transaction

Online	Register	Commit	Vote	Tally
GasCost	157,970	67,761	116,958	49,058
EtherCost	0.0031	0.0013	0.0023	0.0009
USDCost	0.589	0.247	0.437	0.171
Offline	ZKP	Commit	Vote	OrProof
GasCost	317,614	357,151	222,873	2,377,266
EtherCost	0.0064	0.0071	0.0044	0.047
USDCost	1.216	1.349	0.836	0.893

In online phases, it is to verify the proofs in each algorithm. And in offline phases, it is to create a proof in the algorithms.

7.3 Gas Cost on the Blockchain

We also evaluate the algorithms with Ethereum smart contracts written in Solidity[16] on a private blockchain in a test network to test the gas cost. The transactions are deployed with Ethereum wallet [17] with Geth server.[18] The gas cost of each transaction is listed in Table 4. We can see from the Table 4 that, for each transaction, we provide the gas cost as well as the corresponding ether cost and US Dollar cost, in which the gas price is 0.02 Ether per million gas and the Ether is 190 USD (Jul. 2019).[19] Note that, the commitments, the encrypted ballots and the corresponding zero-knowledge proofs can be computed locally. We also provide smart contracts for the aforementioned phases that the users can call to get the values offline. For the online part, Register is a part of Setup, which is to put the public key of a user on the blockchain. Commit and Vote are to put the commitment and the ballots on the blockchain. Register, Commit and Vote all contain the verification of a zero-knowledge proof shown in section 5. Register costs 317,614 gas, which also has several expensive accessing operations to the storage. Vote costs more gas than Commit, as the zero-knowledge proof in the former is more expensive than that in the latter. Tally is the cheapest one, since to tally the result doesn't need zero-knowledge proofs. Besides that, the computation of h_i and Y_i costs 328,454 gas and 319,732 gas respectively for four voters, which equals to 0.0066 ether and 0.0063 ether. There are more reverse operations in h_i than those in Y_i, thus the cost of h_i is a little bit higher than that of Y_i. The USD cost for each online phase is less than 1 dollar, which is acceptable. The offline phase, which costs 1 dollar on average each, is optional.

8 Conclusion

IoT is dramatically changing manufacturing and production in traditional enterprises, which can be combined with blockchain to achieve decentralized IoT. In this paper, we

16. https://solidity.readthedocs.io/en/v0.5.3/
17. https://wallet.ethereum.org/
18. https://geth.ethereum.org/downloads/
19. https://coinmarketcap.com/currencies/ethereum/.

integrated blockchain-based self-tallying voting systems in decentralized IoT architecture to solve the fairness issues in self-tallying systems with two distinct mechanisms and provide a concrete construction. We proved the security of the construction and also implemented it to test the efficiency of the proposed protocol. Future works include designing a prototype of the voting system and constructing a voting protocol for the large universe.

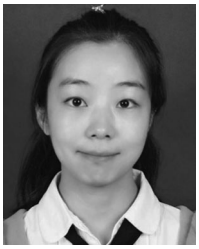
Acknowledgments

This work was supported by National Key R&D Program of China (2017YFB0802000), National Natural Science Foundation of China (61872229, 61960206014), National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20170216) and the Basic Research Program of Qinghai Province (2020-ZJ-701).

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.
[3] H. Al-Hamadi, and I. R. Chen, "Trust-based decision making for health IoT systems," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1408–1419, Oct. 2017.
[4] X. Du, M. Guizani, Y. Xiao, and H. H. Chen, "Defending DoS attacks on broadcast authentication in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 1653–1657.
[5] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proc. 19th Int. Conf. Advanced Commun. Technol.*, 2017, pp. 464–467.
[6] Y. Yu *et al.*, "LRCoin: Leakage-resilient cryptocurrency based on bitcoin for data trading in IoT", *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4702–4710, Jun. 2019.
[7] M. A. Khan and K. Salah," IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, 2018.
[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[9] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT", in *Proc. 2nd Int. Conf. Internet-of-Things Des. Implementation*, 2017, pp. 173–178.
[10] G. Zyskind and O. Nathan, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Security Privacy Workshops*, 2015. pp. 180–184.
[11] X. Huang and X. Du, "Achieving big data privacy via hybrid cloud," in *Proc. IEEE INFOCOM Workshops*, 2014, pp. 512–517.
[12] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 72–83, Feb. 2019.
[13] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.
[14] A. Reyna, C. Martn, J. Chen, E. Soler, and M. Daz, "On blockchain and its integration with IoT," *Challenges Opportunities Future Gener. Comput. Syst.*, vol. 88, pp. 173–190, 2018.
[15] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2017, pp. 618–623.
[16] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, pp. 533–546, 2016. [Online]. Available: http://dx.doi.org/10.4236/jsea.2016.910036
[17] Y. Yu, Y. Li, J. Tian, and J. Liu, "Blockchain-based solutions to security and privacy issues in the Internet of Things," *IEEE Wireless Commun.* vol. 25, no. 6, pp. 12–18, Dec. 2018.
[18] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *Proc. Int. Workshop Public Key Cryptogr.*, 2002, pp. 141–158.
[19] J. Groth, "Efficient maximal privacy in boardroom voting and anonymous broadcast," in *Proc. Int. Conf. Financial Cryptogr.*, 2004, pp. 90–104.
[20] F. Hao, P. Y. A. Ryan, and P. Zielinski, "Anonymous voting by two-round public discussion," *IET Inf. Secur.*, vol. 4, no. 2, pp. 62–67, 2010.
[21] D. Khader, B. Smyth, P. Ryan, and F. Hao, "A fair and robust voting system by broadcast," in *Proc. 5th Int. Conf. Electron. Voting*, 2012, pp. 285–299.
[22] 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.*, 2017, pp. 357–375.
[23] Z. Zhao and T. H. H. Chan, "How to vote privately using Bitcoin," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2015, pp. 82–96.
[24] S. Bistarelli, M. Mantilacci, P. Santancini, and F. Santini, "An end-to-end voting-system based on Bitcoin" in *Proc. Symp. Appl. Comput.*, 2017, pp. 1836–1841.
[25] Y. Li, G. Yang, W. Susilo, Y. Yu, M. H. Au, and D. Liu, "Traceable monero: anonymous cryptocurrency with enhanced accountability," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2910058.
[26] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, 1986, pp. 186–194.
[27] T. Jager, "How to build time-lock encryption," *IACR Cryptol. ePrint Archive*, vol. 2015, no. 478, 2015.
[28] J. Liu, T. Jager, S. A. Kakvi, and B. Warinschi, "How to build time-lock encryption," *Des. Codes Cryptogr.*, vol. 86, pp. 2549–2586, 2018.
[29] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," *Dept. Comput. Sci.*, Dept. Comput. Sci., ETH Zurich, Zürich, Switzerland, Tech. Rep. 260, 1997.
[30] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*. vol. 3, no. 4, pp. 469–472, Jul. 1985.
[31] F. Brandt, "Efficient cryptographic protocol design based on distributed El Gamal encryption," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2005, pp. 32–47.
[32] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. 11th CRYPTO Conf.* 1991, pp. 129–140.
[33] J. Katz, "Digital signatures," Berlin, Germany: Springer, 2010.
[34] Y. Takabatake, D. Kotani, and Y. Okabe, "An anonymous distributed electronic voting system using Zerocoin," *IEICE Techinical Report*. vol. 54, no. 11, pp. 127–131, 2016.
[35] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Proc. Annu. Int. Cryptology Conf.*, 1994, pp. 174–187.
[36] A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung, "On monotone formula closure of SZK," in *Proc. 35th Annu. Symp. Foundations Comput. Sci.*, 1994, pp. 454–465.
[37] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[38] R. L. Rivest, A. Shamir and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996. [Online]. Available: http://bitsavers.trailing-edge.com/pdf/mit/lcs/tr/MIT-LCS-TR-684.pdf
[39] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters, "Time-lock puzzles from randomized encodings," in *Proc. ACM Conf. Innovations Theor. Comput. Sci.*, 2016, pp. 345–356.
[40] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," IACR Cryptology ePrint Archive, vol. 2004, no. 332, 2004.
[41] Y. Li *et al.*, "Toward privacy and regulation in blockchain-based Cryptocurrencies," *IEEE Network*, vol. 33, no. 5, pp. 111–117, Sep./Oct. 2019.
[42] P. K. Sharma, N. Kumar, and J. H. Park, "Blockchain-based distributed framework for automotive industry in a smart city," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2018.2887101.
[43] A. Jindal, G. S. Aujla, and N. Kumar, "SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment," *Comput. Netw.*, vol. 153, pp. 36–48, Apr. 2019.
[44] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *J. Netw. Comput. Appl.*, vol. 126, pp. 45–58, Jan. 2019.
[45] Y. Yu, Y. Zhao, Y. Li, X. Du, L. Wang, and M. Guizani, "Blockchain-based anonymous authentication with selective revocation for smart industrial applications," *IEEE Trans. Ind. Inform.*, vol. 16, no. 5, pp. 3290–3300, May 2020.

[46] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, vol. 99, pp. 173–186, 1999.
[47] V. Santos, J. P. Barraca, and D. Gomes, "Secure decentralized IoT infrastructure," in *Proc. Wireless Days*, 2017, pp. 173–175.

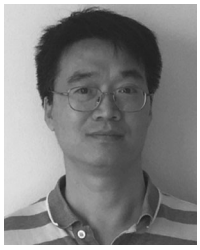


Yannan Li (Student Member, IEEE) received the bachelor's and master's degrees from the University of Electronic Science and Technology of China, in 2017 and 2014, respectively. She is currently working toward the PhD degree with the School of Computing and Information Technology, University of Wollongong, Australia. Her research interests include applied cryptography and cryptocurrencies.

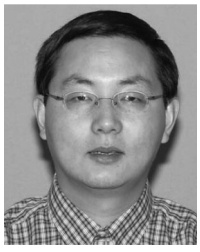
Willy Susilo (Senior Member, IEEE) is a senior professor with the School of Computing and Information Technology, Faculty of Engineering and Information Sciences, University of Wollongong (UOW), Australia. He is the director of Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, UOW and the head of School of Computing and Information Technology at UOW (2015–now). Prior to this role, he was awarded the prestigious Australian Research Council Future Fellowship in 2009. In 2016, he was awarded the "researcher of the Year at UOW, due to his research excellence and contributions. He is the editor-in-chief of the Elsevier's Computers Standards and Interface and the MDPIs Information journal. He is currently an associate editor of the *IEEE Transactions on Dependable and Secure Computing*. He has also served as the program committee member of several international conferences.

Guomin Yang (Senior Member, IEEE) received the PhD degree from the Computer Science Department, City University of Hong Kong in 2009. Formerly, he worked as a Research Scientist in the Temasek Laboratories at National University of Singapore. He is currently an associate professor at the School of Computing and Information Technology, University of Wollongong, Australia. In 2015, he was awarded a prestigious Australian Research Council DECRA fellowship. His research interests include cryptography and network security.

Yong Yu (Member, IEEE) received the PhD degree in cryptography from Xidian University, in 2008. He is currently a professor with Shaanxi Normal University, China. He holds the prestigious one hundred talent Professorship of Shaanxi Province as well. He has authored more than 80 referred journal and conference papers. His research interests include blockchain and cloud security. He is an associate editor of Soft Computing.



Dongxi Liu is a senior research scientist in CSIRO since 2008. His research interests include applied cryptography, data processing, and system security. His recent work aims to design public key encryption based on checkable hardness facts instead of hardness assumptions.



Xiaojiang Du (Fellow, IEEE) is currently a professor with the Department of Computer and Information Sciences, Temple University. He has published more than 400 journals and conference papers in these areas and has been awarded more than $6M research grants from the US National Science Foundation and other agencies. His research interests are security, systems, wireless networks, and computer networks.

Mohsen Guizani (Fellow, IEEE) received the BS (with distinction) and MS degrees in electrical engineering, the MS and PhD degrees in computer engineering from Syracuse University, Syracuse, NY, in 1984, 1986, 1987, and 1990, respectively. He is currently a professor with the Computer Science and Engineering Department, Qatar University, Qatar. Previously, he served in different academic and administrative positions at the University of Idaho, Western Michigan University, University of West Florida, University of Missouri-Kansas City, University of Colorado-Boulder, and Syracuse University. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is currently the editor-in-chief of the *IEEE Network Magazine*, serves on the editorial boards of several international technical journals and the founder and editor-in-chief of Wireless Communications and Mobile Computing journal (Wiley). He is the author of nine books and more than 600 publications in refereed journals and conferences. He guest edited a number of special issues in IEEE journals and magazines. He also served as a member, chair, and general chair of a number of international conferences. Throughout his career, he received three teaching awards and four research awards. He also received the 2017 IEEE Communications Society WTC Recognition Award as well as the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. He is a Senior Member of ACM.