Smart Contract-Based E-Voting System Using Homomorphic Encryption and Zero-Knowledge Proof

Yuxiao Wu and Shoji Kasahara(✉)

Division of Information Science, Nara Institute of Science and Technology, Nara 6300192, Japan
{wu.yuxiao.ws9,kasahara}@is.naist.jp

Abstract. As an indispensable part of establishing modern representative democratic organizations, election is based on a voting process on site or remotely. With the rapid development of information technology, the application of electronic voting systems in practice is significantly increasing in recent years. Consequently, whether an electronic voting system is secure and reliable enough is the most critical factor of the systems. Whereas, most of the existing proposals neglect to confirm the trustworthiness of the administrator, which may impact the security and availability of the system. For this purpose, we propose an up-to-date electronic voting system based on smart contract using additively homomorphic encryption and non-interactive zero-knowledge proof. In our work, we utilize a concise zero-knowledge proof algorithm and an inbound oracle in combination to allow voters to verify the fidelity of the administrator. We prove the feasibility, efficiency, and scalability of our system can satisfy a majority of application scenarios including large-scale voting. In particular, we evaluate the time performance and cost performance and demonstrate its merits including the low cost in many functions and linear performance when generating zero-knowledge proof.

Keywords: Smart contract · Blockchain · E-voting · Zero-knowledge proof · Homomorphic encryption · Oracle

1 Introduction

With the goal to minimize the expense and maximize the efficiency of executing an election, it is becoming increasingly difficult to ignore the security, privacy and compliance when designing electronic voting systems [11]. For this purpose, in the area of e-voting system, there has been a recent surge in interest and research. However, security, transparency, distributed authority, data integrity,

This research was supported in part by Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research (A) No. 19H01103, and Grant-in-Aid for Challenging Research (Exploratory) No. 22K19776.

privacy and compliance requirements have become the main bottlenecks blocking e-voting systems implemented on a large scale [30].

In order to counter the bottlenecks that mentioned above, a promising data structure known as blockchain is initially introduced to e-voting systems. In essence, blockchain is a distributed ledger that has the property of tamper-resistance. That is, the blockchain's database is maintained by each node on chain rather than single node which leads to blockchain be the immutable ledger with transactions in old blocks preserved and transactions in new blocks irreversibly added [19].

With regard to the e-voting systems proposed for implementation using the blockchain technology, a series of requirements and features are summarized to build proper schemes in [28].

1. **Receipt-freeness** [4]**:** Any evidence proving the voter's selection for a particular candidate is not allowed to be revealed.
2. **Fairness** [6,9]**:** Every voter should have the same weight when taking part in the voting.
3. **Data integrity** [12]**:** Each valid vote is recorded correctly and can't be tampered with by any part, once logged.
4. **Privacy** [33]**:** The anonymity of voters should be guaranteed.
5. **Eligibility** [22]**:** The system only accepts the registered voter's ballot.
6. **Reliability** [8]**:** The system is stable and no vote is missed.
7. **Uniqueness** [24]**:** Double voting is not allowed.
8. **Verifiability** [16]**:** Voters have the right to verify whether their ballots are tallied legitimately.

According to the features and requirements that mentioned above, cryptography is considered as one of the most effective technologies that can be utilized to e-voting systems. On one hand, in order to achieve the goal of verifying the trustworthiness of the administrator without information disclosure, zero-knowledge proof is one of the most widely used cryptographic primitives and plays a key role in the field of privacy protection. On the other hand, additively homomorphic encryption has the features that only supports arithmetic for elements of its plaintext space and provides an operation that produces the encryption of the sum of two numbers, given only the encryptions of the numbers [10]. Consequently, it can be effectively applied to e-voting systems that focuses on data confidentiality. In our work, combining these two cryptographic technologies enables us to realize a majority of the above-mentioned requirements relating to information security including receipt-freeness, privacy and verifiability. Whereas, data integrity and reliability relies on the immutability of blockchain and fairness, eligibility and uniqueness is based on the implementation of software system by utilizing solidity and python.

The rest of this paper is organized as follows. Section 2 presents related work and Sect. 3 introduces some fundamental technologies as preliminaries. Section 4 represents our proposed e-voting system, and Sect. 5 shows the performance evaluation of our system from the perspective of time performance and gas fee performance. Finally, Sect. 6 concludes the paper.

2 Related Work

The existing e-voting schemes that blockchain is applied to are classified into two types: blockchain-based e-voting system and e-voting system using blockchain.

2.1 Blockchain-Based E-Voting System

In the blockchain-based e-voting system, the whole system is built upon the blockchain framework. In other words, blockchain is convinced to be the infrastructure that sustains the system. Most of the blockchain-based e-voting system adopt the permissioned blockchain structure instead of permissionless blockchains such as Bitcoin and Ethereum.

In [11], the authors propose a blockchain-based e-voting system based on a permissioned blockchain, and some of the popular blockchain frameworks are evaluated for constructing a blockchain-based e-voting system. Sun et al. propose a simple voting protocol based on the existing Quantum Blockchain [13,25]. It satisfies the most important properties of secure voting protocols by introducing matrix and number theory to the stages of ballot commitment and ballot tallying [24]. In addition, a blockchain-enabled large-scale e-voting system with robustness and universal verifiability is presented in [32]. In this system, a hybrid approach which combines the counting bloom filter and Merkle hash tree is developed in order to break the bottleneck of cost performance.

Generally speaking, this kind of e-voting systems has the merits of cost-friendly performance and high response rate. However, a major problem with this approach is utilizing the permissioned blockchain as the blockchain framework of the system, leading to lower reliability for voters. In particular, most of the nodes in the system are maintained by the administrator, making it more difficult for voters to verify the fidelity of the administrator. This verification by voters is one of our goals.

2.2 E-Voting System Using Blockchain

Referring to the relation between Infrastructure-as-a-service (Iaas) and Platform-as-a-service (Paas) (two kinds of cloud service delivery models) in cloud computing [5], the blockchain in this kind of e-voting system acts as the platform rather than the infrastructure compared with that in blockchain-based e-voting systems. For this reason, this kind of systems mainly focuses on smart contract provided by the blockchain. Consequently, Ethereum is convinced to be one of the most suitable blockchain frameworks for this kind of systems due to its reliability and functionality.

The authors of [31] present a platform-independent secure and verifiable voting system combining with a variety of cryptographic techniques including Paillier cryptosystem that supports the execution of a smart contract. In [18], Lyu et al. propose a trustless e-voting system, which is deployed on Ethereum by smart contract. The authors use linkable ring signature [7,14,15] and threshold encryption without a trusted third party in their system. Except for these e-voting systems focusing on security and privacy by cryptography, some other systems concentrating on the countermeasure for the low efficiency and expensive cost also utilize blockchain as the platform. Based on web3 framework and POA network, Al-Madani et al. make their system to support real-time service without using any cryptosystem [3].

In contrast to blockchain-based e-voting systems, e-voting systems using blockchain are commonly doubted due to the higher expense. However, this kind of systems has great reliability because of introducing the existing famous blockchain framework (e.g. Ethereum). For this reason, we design our system based on the criterion that is similar to e-voting systems using blockchain.

3 Preliminaries

3.1 Smart Contract

In order to minimize contracting cost between transacting parties and to avoid accidental exceptions or malicious actions during contract performance, Nick Szabo suggested translating the clauses of a contract into code and embedding them into software or hardware to make them self-execute, which is known as smart contract [27,34].

As a program running on a blockchain, a smart contract can be correctly executed by a network of mutually distrusting nodes without the need of an external trusted authority. The self-executing nature of smart contracts provides a tremendous opportunity for use in many fields that rely on data to drive transactions [26].

Blockchains can be divided into permissioned blockchains (i.e. non-public) and permissionless blockchains (i.e. public). Permissionless blockchain platforms allow any user to join the network while permissioned blockchain platforms allow only permitted users to join [34]. Different blockchain platforms provide different support for smart contracts. Some (e.g. Bitcoin) may only allow users to use a simple scripting language to develop smart contracts with simple logic; while some platforms, such as Ethereum, support much more advanced programming languages for writing smart contracts [23].

3.2 Homomorphic Encryption

An encryption is homomorphic if from Enc(a) and Enc(b), it is possible to compute Enc(f(a, b)) where Enc(x) represents the encryption of x and f is $+$, $\times$, or $\oplus$ (exclusive OR) and without using the private key for decryption [29].

Homomorphic encryption can be categorized into three types of schemes with respect to the number of allowed operations on the encrypted data; (1) Partially Homomorphic Encryption (PHE), which allows only one type of operation with an unlimited number of times (i.e., no bound on the number of usages), (2) Somewhat Homomorphic Encryption (SWHE), which allows some types of operations a limited number of times, and. (3) Fully Homomorphic Encryption

(FHE), which allows an unlimited number of operations for an unlimited number of times [1].

Paillier encryption is classified as Partially Homomorphic Encryption. In detail, it is an additively homomorphic encryption scheme based on composite degree residuosity classes. Paillier encryption is provably secure under appropriate assumptions in the standard model [20]. Simultaneously, Paillier encryption is proved to have the additive homomorphism so that it can be appropriately applied to e-voting systems because of the massive utilizing of add operation in these systems.

3.3 Zero-Knowledge Proof

Roughly speaking, the zero-knowledge proof realizes a scenario that a prover who wants to convince a verifier that some statement is true without revealing any other information.

As the applications, privacy-preserving systems use zero-knowledge proofs to prove the correctness of outputs without revealing sensitive information about inputs. In online voting systems, voters prove that they correctly encrypted their vote, without revealing any information about the selected candidate [2].

The authors of [17] propose zksk, a well-documented Python library for defining and computing sigma protocols (the most popular class of zero-knowledge proofs). In zksk, smaller proofs can be converted into building blocks that then can be combined into bigger proofs. In addition, compared with the large size of key and proof of traditional zero-knowledge proofs such as zk-SNARKs, zksk is more practical in some areas due to its smaller data size as low as several bytes.

4 Proposed System

This section describes the proposed smart contract-based e-voting system using partially homomorphic encryption and non-interactive zero-knowledge proof. Furthermore, oracle is another crucial technology applied to this system so that data privacy in the Ethereum-based smart contract is feasible to be guaranteed against external homomorphism attackers and internal untrusted entities.

4.1 System Components

The roles of users that participate the voting system are classified into three types: administrator, voter, and observer. The administrator and voter are indispensable for the voting system and have their own nodes, whereas the observer is optional and is not compulsively required to have a node in Ethereum. Moreover, the voter who tries to verify the zero-knowledge proofs associated with other voters is also regarded as an observer. The definitions of the three roles are as follows.

1. **Administrator:** The entity which is set up to manage and maintain the whole process of voting. The administrator's work includes managing voter registration, initializing voter identity, creating smart contracts for the election, verifying ballots from voters, generating zero-knowledge proof, transmitting processed data to the oracle node, and announcing the voting result.
2. **Voter:** The entity which is eligible to act as a participant in the democratic election process. Valid ballots from voters are the essential component that results in formal voting. It is feasible for voters to register in the system using their identifiers such as Ethereum addresses, request the public key published in the blockchain, cast ballots in the smart contract, and verify the corresponding zero-knowledge proof which reveals their own identities.
3. **Observer:** The entity which is an extra part of the system roles to assure the feasibility and availability of the system. Observers are required to verify the zero-knowledge proofs of each voter and publish the verification result individually so that the public can decide whether the result published by the administrator is trusted or not.

We adopt smart contracts for implementing our voting system. The system has three types of smart contracts: Contract Voting, Contract VID, and Contract ZKP. The Contract Voting provides the functions of casting the ballot and processing the original voting data. Contract VID is used for publishing all voter IDs. Contract ZKP provides the functions of zero-knowledge proof for observers and voters. The details of the smart contracts are as follows.

1. **Contract Voting:** The Contract Voting manages the voters' identifiers and announces the administrator's public key to the eligible voters. The voters' identifiers are registered in the address array type named "Alist". The administrator verifies the eligibility of a voter by checking his/her identifier in Alist. If the administrator authenticates the voter as an eligible voter, the administrator announces his/her public key to the voter with a function of "announce_PK". When the voter casts a ballot, the administrator verifies his/her voting data with the function of "verify_ballot".
2. **Contract VID:** The Contract VID manages the identifiers of the voters whose ballots are verified to be valid and publishes them without revealing voters' privacy. All voters' identifiers are stored in a string array type called "VID_list". At the stage of result announcing, any voter and observer can call the function named "return_VID" to acquire all voters' identifiers.
3. **Contract ZKP:** The Contract ZKP manages the zero-knowledge proofs according to ballots that have been verified to be valid and allows the voter to retrieve his/her corresponding proof and verification key. A parameter in the type of string of "ZKP" is utilized to request zero-knowledge proof data downloaded from the server via oracle. When a voter wants to search the corresponding zero-knowledge proof, he/she can get the result with a function known as "search_ZKP" as long as the valid address is provided.

The data format of a vote is the encoding of its ballot. Due to the application of partially homomorphic encryption in this system, the form of voting data is designed to be compatible with it. After generating his/her voting data of which detail will be described in the following part, each voter encrypts the voting data with the public key of the administrator using Paillier encryption.

Let M and N denote the number of candidates and that of voters, respectively. We define the following variables for encoding.

- We define Voted_ballot(j) ($j \in \{1, \ldots, N\}$) as voter j's selection among candidates. If voter j votes for candidate $i \in \{1, \ldots, M\}$, Voted_ballot(j) is set to
$$\text{Voted\_ballot}(j) = (10^8)^{i-1}.$$
For example, consider candidates A and B are indexed with 1 and 2, respectively. If voter j votes for candidate A (resp. B), Voted_ballot(j) is set to 1 (resp. 100,000,000). Here, we set the interval of 10^8 for balancing the data size and the scalability.
- Voter_address(j) is the voter j's address in the blockchain network. For example, in Ethereum, if voter j's address is 0xeC2804Dd9B992C10396b5Af176f06 923d984D90e, this value is substituted to Voter_address(j).
- Each voter has his/her own unique voter ID which is generated by the administrator at the voter registration stage. Let Voter_ID(j) denote the ID of voter j. The administrator randomly generates a number from 1 to 99,999,999 for each voter who passed the identity verification and distributes it to the corresponding voter privately. Consider the case of 100,000,000 voter's participation. Assuming the extreme situation that all of them vote for A, the resulting sum of the Voted_ballot is 100,000,000. Under this condition, we are unable to distinguish the situations that whether there is one voter voting for B or there are 100,000,000 voters voting for A except for additional verification. In order to avoid these matters, we set the maximum capacity of voters to be 99,999,999 and correspondingly make sure that all of voter's ID are in the range from 1 to 99,999,999. Each voter knows his/her own voter ID, and the administrator can check the mapping relation between the voter's address and voter ID.
- At voting, each voter generates his/her own voting data. We define Voting_data(j) as the voting data of voter j, which is given by
$$\text{Voting\_data}(j) = \\ \text{Voted\_ballot}(j) + \text{Voter\_address}(j) + \text{Voter\_ID}(j).$$
Each voter calculates his/her Voting_data, encrypts the Voting_data as the ballot and sends it to the Contract Voting for voting.

The voting data processing for the administrator is classified into three task types. The first task type is retrieving the voting data array and address array from Contract Voting, then the administrator will rebuild these two arrays and generate an aggregated list consisting of the voter ID, voting data, and zero-knowledge proof. The second task type is verifying the validity of each ballot and generating the zero-knowledge proof for the voters whose ballots are verified to be valid. Last but not least, the list containing information processed in the former two steps is uploaded to a web page in json type deployed in our server built up for the oracle node to achieve data.

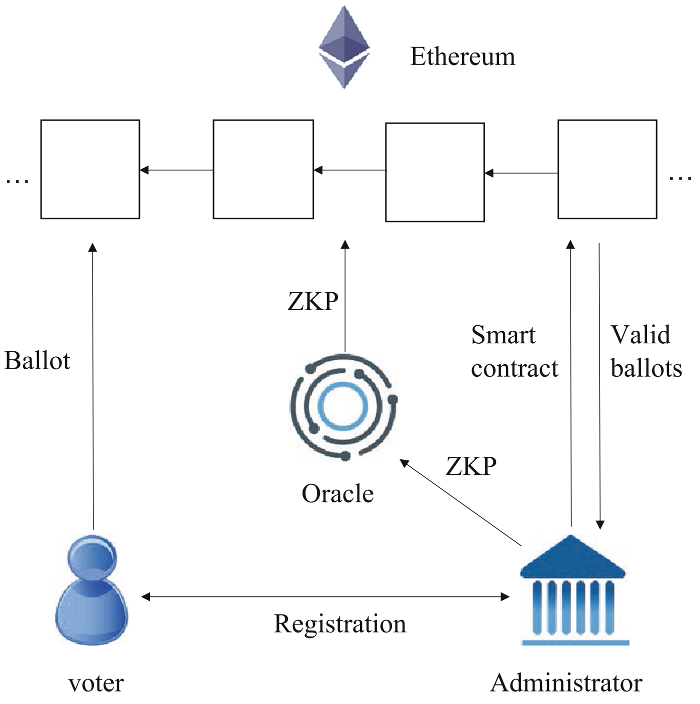

Fig. 1. Overview of the System.

4.2 System Model

Figure 1 illustrates the overview of this system. The procedures included in our system are (1) voter registration, (2) system initialization, (3) contract creation, (4) voting, (5) ballot verification, (6) ballot tallying and result announcing, (7) zero-knowledge proof verification, (8) trustworthiness verification. The functions of each procedure are as follows.

1. **Voter registration:** The registration is held by the administrator and aims to sign up the eligible users and distribute the voter IDs. In detail, the administrator can authenticate the voter identity in two methods in accordance with the authentication site: on-site authentication and online authentication. In on-site authentication, the administrator directly authenticate the valid identity certificate of participants face to face. As soon as authentication finished, the administrator informs the corresponding voter ID to the voter. In the

other case, with the help of technologies including blockchain and encryption, the system assures that the voter's identity is authenticated and the voter ID is distributed in a private channel which is independent of the blockchain.

2. **System initialization:** The administrator processes the information received in voter registration and transforms it into arrays that can be used in the following procedures. On the one hand, the administrator creates an "Alist" containing all voters' addresses and uploads it to contract Voting. On the other hand, the administrator generates a mapping list called "Mlist" that maps the voter's address to the voter ID according to voter registration, and saves it privately and locally.
3. **Contract creation:** In this step, the smart contracts mentioned in Subsect. 4.1 are instantiated and deployed to Ethereum. Except for the instantiation and deployment, the administrator collects the addresses of these contracts from Ethereum and publishes them to voters.
4. **Voting:** Voting is the interactive process between contract Voting and voters, and is considered as a core operation in this system. In advance, voters are required to calculate their voting data in plaintext. Then, without the participation of the administrator, the voter calls the function "announce_PK" to obtain the administrator's public key based on Paillier encryption. Furthermore, voters encrypt the voting data and upload them to contract Voting.
5. **Ballot verification:** Ballot verification is comprised of two stages: verification by contract and verification by the administrator. When contract Voting receives the encrypted voting data, it automatically verifies whether the address that the data come from is registered and whether there is any voter who casts the ballot more than one time. The ballots that do not satisfy the two conditions are discarded. The ballots that are not discarded in the first stage are then verified by the administrator for the correctness of the original voting data. The administrator decrypts the voting data in ciphertext and judges the decrypted voting data using "Mlist". Ballots satisfying the conditions in both two stages are judged as valid.
6. **Ballot tallying and result announcing:** When the process of voting ends, the administrator collects all valid ballots and adds their voting data in ciphertext together to calculate m_{sum}, which is given by

$$m_{sum} = \sum_{j \in \mathcal{N}_{vb}} \text{Voting\_data}(j),$$

where $\mathcal{N}_{vb}$ is the index set of the voters whose ballots are judged as valid. Then, the administrator decrypts m_{sum} and gains the result which can represent the voting result by the following formula

$$\begin{aligned} &\text{Result} = \\ &m_{sum} - \sum_{j \in \mathcal{N}_{vb}} \{\text{Voter\_address}(j) + \text{Voter\_ID}(j)\}\,. \end{aligned}$$

Then, the administrator conducts vote counting and announces the winner on the official platform.

7. **Zero-knowledge proof verification:** In this system, we apply a non-interactive zero-knowledge proof proposal called "zksk" [17] as the method of zero-knowledge proof. The proof is generated just after the ballot is verified as valid at both the two stages. Here, each voter's ID is kept secret. The detailed operations of the prover (the administrator) is illustrated in Algorithm 1.

Algorithm 1. Generating Zero-knowledge Proof (operations on the prover's side).

```
Input: group($G$,$g$), voters' identifiers VID, random values $n$, $r$, $k$
Output: proof $\pi$, verification key $y$
1: $secret = \text{VID}$
2: $y = secret \times g$
3: $stmt = (y, n \times g)$
4: $pre\_com = \text{None}$
5: $com = r$
6: $chal = H(y \mid\mid pre\_com \mid\mid com)$
7: $resp = k + secret \times chal$
8: $\pi = (pre\_com, chal, resp)$
9: return $\pi$, $y$
```

When the zero-knowledge proof has been generated according to the above algorithm, the administrator uploads it to contract ZKP which is mapped to the voter's address. The voter or any observer can access the proof with the specific address through the calling of contract ZKP. Then, they conduct Algorithm 2 to verify the proof.

Algorithm 2. Verifying Zero-knowledge Proof (operations on the verifier's side).

```
Input: group($G$,$g$), proof $\pi$, verification key $y$
Output: proving result True/False
1: $new\_com = resp \times g + (-chal) \times y$
2: $new\_chal = H(y \mid\mid pre\_com \mid\mid com)$
3: if $new\_chal == chal$ then
4:     return True
5: else
6:     return False
7: end if
```

If a voter verifies the proof associated with him/her successfully, he/she can trust that his/her ballot is tallied correctly. On the other hand, an observer can verify the proof of any address to check whether there is any extra ballot falsified by the administrator.

8. **Trustworthiness verification:** The procedure of trustworthiness verification is executed to verify the fidelity of the administrator from a different

aspect with zero-knowledge proof verification. The observer utilizes m_{sum} in procedure 6 and does reverse operations to check whether the administrator revised the result. The verification process is shown in Algorithm 3.

Algorithm 3. Trustworthiness Verification.

```
Input: voting result Result, the sum of voting data $m_{sum}$, public key $(n_1, g_1)$, voters'
    addresses Voter_address, voters' identifiers Voter_ID
Output: verification result True/False
 1: $m = \text{Result} + \sum_{j \in \mathcal{N}_{vb}} \{\text{Voter\_address}(j) + \text{Voter\_ID}(j)\}$
 2: $\exists\ r,\ m_{sum} == {g_1}^m \times r^{n_1} \mod {n_1}^2$
 3: $\exists\ r, s,\ m_{sum} + s \times {n_1}^2 == {g_1}^m \times r^{n_1}$
 4: Condition 1: $\exists\ s, (m_{sum} + s \times {n_1}^2 \mod {g_1}^m) == 0$
 5: Condition 2: $\exists\ s\ s.t.\ Condition1,\ \sqrt[n_1]{\frac{m_{sum} + s \times {n_1}^2}{{g_1}^m}} \in \mathcal{N}$
 6: if Condition 1 and Condition 2 are True then
 7:     return True
 8: else
 9:     return False
10: end if
```

4.3 System Features

Based on the system components and system model, a slice of features are clarified. In our work, one of the most popular permissionless blockchains known as Ethereum is applied as the blockchain framework of the system so that reliability and robustness of the system is ensured by the stability of Ethereum and smart contract themselves.

As mentioned in Subsect. 4.2, we utilize a succinct zero-knowledge proof known as "zksk". In comparison with other non-interactive zero-knowledge proofs (e.g. zk-SNARK), the size of zksk's proving key and verification key is smaller and can be recorded in a parameter in string type. Thus, zksk is firmly convinced to be extremely compatible with this system rather than many other non-interactive zero-knowledge proofs due to the increasing cost according to data size when using oracle. Consequently, the application of zksk confirms the verifiability ensuring the voter's right to learn that his/her ballot has been correctly tallied.

The voter's privacy is assured by utilizing Paillier encryption, which is also known as an additively homomorphic encryption algorithm. The correctness of the formula that used to check whether the administrator revises the result is based on the additively homomorphism of Paillier cryptosystem.

In addition, receipt-freeness, fairness, data integrity, and uniqueness are ensured by the system structure which has been introduced in Subsect. 4.2. In detail, receipt-freeness and fairness are decided by the procedures including voter registration, voting, and ballot verification whereas data integrity and uniqueness are based on the verification process executed in ballot verification procedure.

5 Performance Evaluation

5.1 Experiment Environment

As explained in Subsect. 4.3, we apply Ethereum to the development of the proposed system, which is a permissionless blockchain as the blockchain infrastructure. Due to the cost saving, we utilize a local Ethereum blockchain known as Ganache to simulate transactions on Ethereum.

In our work, we implement the system in Python and Solidity. The functions described in Subsect. 4.1 are written in solidity. The remaining functions including zero-knowledge generation, server command, the second stage of ballot verification are realized by Python.

In order to enhance the connection between on-chain environment and off-chain environment and to upload zero-knowledge proof to contract ZKP, we select Provable [21] as the oracle in our system. With the merit of low cost and high response speed, Provable is feasible when applied in our system. In Ganache experiment environment, we set up a Ethereum bridge that allows Provable to be utilized to the testnet.

For evaluating the performance of the system, we run voter client, administrator client, and ganache client in the PC with Intel Pentium 4415U CPU, NVIDIA GeForce 940MX GPU, and ADATA DDR4 2666 12G RAM.

5.2 Execution Time Performance

In this subsection, we investigate the component-wise execution-time performance of the proposed voting system in order to figure out the time consumption of the whole system and which function takes over most of the time.

The functions measured in this experiment are: loading ZKP, loading VID, generating ZKP, processing data on administrator side, voting (successful), voting (double voting), voting (not in Alist), encrypting data, and decrypting data. Table 1 illustrates the execution time performance of each function under the condition of one voter's participation.

In Table 1, loading ZKP takes over most of the time, which exceeds 50% of the whole executing time. Following loading ZKP, the execution times of loading VID and generating key are around 2 s. The execution time of voting in case of a valid ballot is over 1 s, while the execution times of voting in cases of double voting and not-in-Alist are smaller than that of valid voting. The reason for this result is that loading operation in smart contract needs to send VIDs and zero-knowledge proofs from the off-chain environment to the on-chain environment using oracle. Therefore, it takes a lot of time for data exchange among different environments. Meanwhile, generating keys and voting also take more time than other operations due to the complicated cryptographic algorithm

Table 1. Execution time performance.

Operation	Time [s]
Generate Key	2.016
Encrypt data	0.054
Decrypt data	0.105
Vote (successful)	1.312
Vote (double voting)	0.516
Vote (not in Alist)	0.797
Process Data on Admin Side	0.787
Generate ZKP	0.003
Load VID	2.224
Load ZKP	9.226

in them. According to the analysis, we have two conclusions. First of all, the whole execution time when casting per ballot is about 1.438 s to 2.243 s, which is better than a majority of existing systems that have been investigated on. What's more, the operations executed by smart contract are the main reasons that result in low efficiency.

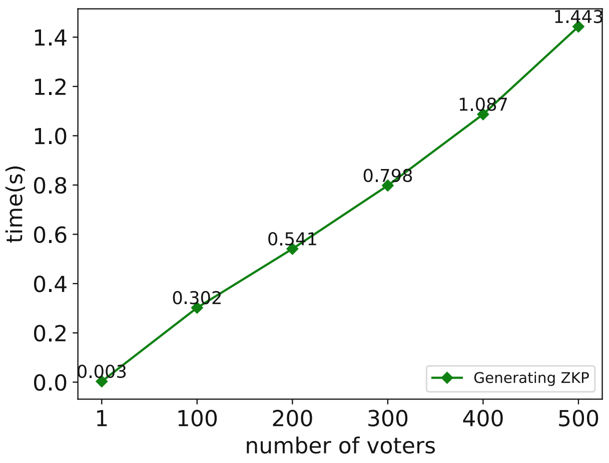

Fig. 2. Generating Zero-knowledge Proof.

Generating zero-knowledge proof is regarded as the most core part of this system. Figure 2 demonstrates the total execution time for generating zero-knowledge proof in cases of the number of voters participating the system equal to 1, 100, 200, 300, 400 and 500. With up to 500 voters' participation, our system has the ability to support zero-knowledge generating within 1.5 s, which is fast enough for a large-scale election at the present stage. Meanwhile, this experiment proves that the proposed voting system exhibits good scalability performance since the time for generating zero-knowledge proof is proportional to the number of voters.

5.3 Gas Fee

In this subsection, we investigate how much gas fee of Ethereum is needed for the proposed voting system. We measure gas fee of the following functions: deploying contract ZKP, deploying contract VID, announcing public key, verifying ballot, achieving address list, achieving data list, returning VID, oracle operation, and receiving ZKP. Table 2 shows the above-mentioned functions' gas fees.

Table 2. Gas Fee Performance.

Operation	Gas Fee (gas)
Deploy voting.sol	151305
Deploy VID.sol	375005
Deploy ZKP.sol	2043210
Announce Public Key	0
Verify Ballot	2796389
Achieve Address List	0
Achieve Data List	0
Return VID	1382626
Utilize Oracle	1623253
Receive ZKP	278617

We observe in Table 2 that operations including announcing public key, achieving address list, and achieving data list are free of charge. The functions designed such that those are executed without revision by marking them with "view" in the definition statements of the functions when programming lead to this result. For this reason, plenty of gas fee is saved. The execution of verifying ballot consumes almost 0.007 Ether of gas fee, which is spent more than other operations. The utilizing of multiply iterations is the cause resulting in the excessive consumption of gas fee when executing the operation of verifying ballot.

Adding all gas fees in Table 2 yields the total cost of the operations under the condition of one voter's participation equal to about 0.027 Ether. From another perspective, by employing qualitative modes of enquiry, we attempt to illuminate that the functions regarding contract deployment and data comparison take over a majority of expense of this system. The strategy on cost saving of the system can be established based on these functions.

6 Conclusion

In this paper, we proposed a smart contract-based e-voting system, which can satisfy most of the requirements, especially verifiability. Through the experiments on the execution-time performance and gas consumption, we showed the feasibility, efficiency, and scalability of our system, that can satisfy a majority of application scenarios including large-scale voting.

In future work, we consider the following aspects.

– We have developed a system that combines multitudinous state-of-the-art techniques to make sure the data integrity and reliability, proving its performance based on qualitative evaluation. However, there is increasing concern that our system is not so stable as our evaluation because of the partial execution of centralized operations in the system. In particular, some of the voter's privacy is possibly disclosed if the administrator has the malicious intention in these operations. According to these factors, we shall focus on how to establish a more decentralized system and to take smart contract as the replacement of the administrator in more functions.
– Through the analysis of the execution-time performance and gas fee, we found that several functions can be modified for building a more cost-saving and efficient system. For this purpose, not only in python, but also in solidity, we shall apply some new strategies to the functions. In one case, as we mentioned, the operation of loading ZKP takes over 50% of the whole execution time, which affects the efficiency of the system. Therefore, our strategy for enhancing the execution time from this perspective is to change the oracle that has a better response performance. Consequently, designing and implementing these strategies are another core points in our future work.

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. **51**(4), 1–35 (2018). https://doi.org/10.1145/3214303
2. Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.J., et al.: Electing a university president using open-audit voting: analysis of real-world use of Helios. EVT/WOTE **9**(10) (2009)
3. Al-madani, A.M., Gaikwad, A.T., Mahale, V., Ahmed, Z.A.: Decentralized E-voting system based on smart contract by using blockchain technology. In: 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), pp. 176–180 (2020). https://doi.org/10.1109/ICSIDEMPC49020.2020.9299581
4. Ali, S.T., Murray, J.: An overview of end-to-end verifiable voting systems. In: Real-World Electronic Voting, pp. 189–234 (2016)
5. Almorsy, M., Grundy, J., Müller, I.: An analysis of the cloud computing security problem (2016). https://doi.org/10.48550/ARXIV.1609.01107, https://arxiv.org/abs/1609.01107

6. Anane, R., Freeland, R., Theodoropoulos, G.: e-voting requirements and implementation. In: The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), pp. 382–392 (2007). https://doi.org/10.1109/CEC-EEE.2007.42
7. Au, M.H., Liu, J.K., Yuen, T.H., Wong, D.S.: ID-based ring signature scheme secure in the standard model. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 1–16. Springer, Heidelberg (2006). https://doi.org/10.1007/11908739_1
8. Bokslag, W., de Vries, M.: Evaluating e-voting: theory and practice. CoRR abs/1602.02509 (2016). https://arxiv.org/abs/1602.02509
9. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_66
10. Hardy, S., et al.: Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. CoRR abs/1711.10677 (2017). https://arxiv.org/abs/1711.10677
11. Hjálmarsson, F.ß., Hreióarsson, G.K., Hamdaqa, M., Hjálmtýsson, G.: Blockchain-based e-voting system. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 983–986 (2018). https://doi.org/10.1109/CLOUD.2018.00151
12. Keshk, A.E., Abdul-Kader, H.M.: Development of remotely secure E-voting system. In: 2007 ITI 5th International Conference on Information and Communications Technology, pp. 235–243 (2007). https://doi.org/10.1109/ITICT.2007.4475655
13. Kiktenko, E.O., et al.: Quantum-secured blockchain. Quantum Sci. Technol. **3**(3), 035004 (2018). https://doi.org/10.1088/2058-9565/aabc6b, https://dx.doi.org/10.1088/2058-9565/aabc6b
14. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_28
15. Liu, J.K., Wong, D.S.: Linkable ring signatures: security models and new schemes. In: Gervasi, O., et al. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005). https://doi.org/10.1007/11424826_65
16. Liu, Y., Wang, Q.: An E-voting protocol based on blockchain. Cryptology ePrint Archive, Paper 2017/1043 (2017). https://eprint.iacr.org/2017/1043
17. Lueks, W., Kulynych, B., Fasquelle, J., Bail-Collet, S.L., Troncoso, C.: zksk: a library for composable zero-knowledge proofs. In: Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society (WPES@CCS), pp. 50–54 (2019)
18. Lyu, J., Jiang, Z.L., Wang, X., Nong, Z., Au, M.H., Fang, J.: A secure decentralized trustless E-voting system based on smart contract. In: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 570–577 (2019). https://doi.org/10.1109/TrustCom/BigDataSE.2019.00082
19. Pahlajani, S., Kshirsagar, A., Pachghare, V.: Survey on private blockchain consensus algorithms. In: 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), pp. 1–6 (2019). https://doi.org/10.1109/ICIICT1.2019.8741353

20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-x_16
21. Provable: Provable documentation. https://docs.provable.xyz/
22. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt á voter: a voter-verifiable voting system. IEEE Trans. Inf. Forensics Secur. **4**(4), 662–673 (2009). https://doi.org/10.1109/TIFS.2009.2033233
23. Seijas, P.L., Thompson, S., McAdams, D.: Scripting smart contracts for distributed ledger technology. Cryptology ePrint Archive, Paper 2016/1156 (2016). https://eprint.iacr.org/2016/1156
24. Sun, X., Wang, Q., Kulicki, P., Sopek, M.: A simple voting protocol on quantum blockchain. Int. J. Theor. Phys. **58**(1), 275–281 (2019)
25. Sun, X., Wang, Q., Kulicki, P., Zhao, X.: Quantum-enhanced logic-based blockchain I: quantum honest-success byzantine agreement and qulogicoin (2018). https://doi.org/10.48550/ARXIV.1805.06768, https://arxiv.org/abs/1805.06768
26. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc. (2015)
27. Szabo, N.: Formalizing and securing relationships on public networks. First Monday (1997)
28. Taą, R., Tanrłöver, Ö.Ö: A systematic review of challenges and opportunities of blockchain for E-voting. Symmetry **12**(8) (2020). https://doi.org/10.3390/sym12081328, https://www.mdpi.com/2073-8994/12/8/1328
29. Tebaa, M., Hajji, S.E., Ghazi, A.E.: Homomorphic encryption method applied to cloud computing. In: 2012 National Days of Network Security and Systems, pp. 86–89 (2012). https://doi.org/10.1109/JNS2.2012.6249248
30. Vivek, S., Yashank, R., Prashanth, Y., Yashas, N., Namratha, M.: E-voting systems using blockchain: an exploratory literature survey. In: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 890–895 (2020). https://doi.org/10.1109/ICIRCA48905.2020.9183185
31. Yu, B., Liu, J.K., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P., Au, M.H.: Platform-independent secure blockchain-based voting system. In: Chen, L., Manulis, M., Schneider, S. (eds.) ISC 2018. LNCS, vol. 11060, pp. 369–386. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99136-8_20
32. Zhang, S., Wang, L., Xiong, H.: Chaintegrity: blockchain-enabled large-scale E-voting system with robustness and universal verifiability. Int. J. Inf. Secur. **19**(3), 323–341 (2020)
33. Zhang, W., et al.: A privacy-preserving voting protocol on blockchain. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 401–408 (2018). DOI: https://doi.org/10.1109/CLOUD.2018.00057
34. Zou, W., et al.: Smart contract development: challenges and opportunities. IEEE Trans. Softw. Eng. **47**(10), 2084–2106 (2021). https://doi.org/10.1109/TSE.2019.2942301