

Performance of the Secret Electronic Voting Scheme Using Hyperledger Fabric Permissioned Blockchain

Denis Kirillov(✉), Vladimir Korkhov, Vadim Petrunin, and Mikhail Makarov

Saint-Petersburg State University, Saint Petersburg, Russia
{d.kirillov,v.korkhov,v.petrunin}@spbu.ru
makarovmma@gmail.com

Abstract. Issues related to reliable electronic voting are still very relevant and not fully resolved. The use of distributed ledger technologies for these purposes has great potential, but even private blockchain solutions often have insufficient bandwidth to satisfy the needs of at least corporate voting, not to mention the public elections. The article studies one of such voting systems, which is based on Hyperledger Fabric permissioned blockchain, where the Identity Mixer technology is used as the anonymizing part.

Keywords: E-voting · Blockchain · Hyperledger fabric · Distributed ledger technologies · Performance

1 Introduction

Attempts to use distributed ledger technologies in various areas of our lives are not always successful. We can observe quite successful cases, for example, the Russian IPChain [1], which allows users to register and perform operations with intellectual property objects. Here, according to the developers, the throughput of about 100 transactions per second is enough, and Hyperledger Fabric [11] can provide this. However, along with this, there are many examples in which the blockchain is used because it is fashionable, despite the fact that there are perfectly functioning examples without using distributed ledger technologies.

The same is observed in the field of electronic voting. Despite the fact that there may be solutions that satisfy many (if not all) of the requirements of an ideal system:

- Eligibility,
- Un-reusability,
- Un-traceability,
- Verifiability,
- Receipt-freeness.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12251, pp. 25–36, 2020.
https://doi.org/10.1007/978-3-030-58808-3_3

there remains one significant performance issue. Most modern systems cannot provide transaction processing speeds of more than a couple of tens of thousands per second (NEO [2], EOS [3]), and the most popular (Bitcoin [4], Ethereum [5]) are only a few tens per second.

In addition to this problem, there are at least two more:

- Complexity,
- Lack of standards.

And if the first one has the consequence of the high cost of the implemented solution, the slowness of development, etc., the second one requires a more global approach, and cannot be resolved momentarily.

The rest of this paper is organized as follows. Section 2 reviews the articles related to this topic. An electronic voting protocol using distributed ledger technologies, as well as a study of its performance, are presented in Sect. 3. Section 4 describes a theoretical modification of the previous protocol, in which the blockchain can be replaced by the more lightweight Identity Mixer technology. Section 5 concludes the paper.

2 Related Work

Historically, electronic voting along with supply chains is one of the most common blockchain cases. Often, electronic voting even appears in tutorials on various platforms of distributed ledger as a basic project for studying technology. This fact strongly affected the modern trends of the scientific developments in the field of blockchain technologies. As a result, today there are a huge number of scientific articles on electronic voting. While some focus on the voting protocol in order to provide basic voting requirements, others set out to explore alternative ways of anonymizing in the system or to study performance.

Most existing solutions are based on the use of well-known DLT platforms, but there are also a large number of works where the authors, inspired by the main advantages provided by the blockchain, create products without using existing platforms. The article [6] describes the principle of operation of the created electronic voting system without using ready-made blockchain platforms. The implemented solution is based on the NodeJS-ReactJS bundle. Distribution is achieved by expanding the various nodes based on Heroku, interacting with PubNub - the implementation of publisher-subscribe. The article describes an example of a system deployment for the case of geographically distributed voting, when each block is deployed on a separate blockchain network node. The described system is a traditional implementation of the public blockchain, the maintenance of which is ensured by supporting the possibility of mining, that is, the basic principles of the PoW algorithm are used. The mathematical PoW problem to be solved is also based on the principle of one-way functions, and the algorithm for adjusting the complexity of the task is similar to the principle of Bitcoin functioning. The authors set a reduction in the amount of stored data as one of the objectives of the study; as a result, the system uses Elliptic-curve

cryptography (ECC), which provides a smaller key size, thereby reducing the amount of data storage compared to RSA-based systems.

An example of the implementation of yet another electronic voting system without using DLT platforms was described in [7]. In addition, ECC is also used here. The system is implemented under Ubuntu OS using the Python programming language. Like most systems, the system is designed to meet the basic requirements for electronic voting, including support for the possibility of re-voting until the voting deadline is reached. From an architectural point of view, the implemented system is a modular system and consists of three independent components:

- Synchronized model of voting records based on DLT to avoid forgery of votes,
- User credential model based on elliptic curve cryptography (ECC) to provide authentication and non-repudiation,
- Withdrawal model that allows voters to change their vote before a preset deadline.

For most blockchain applications, the hierarchical principle of maintaining a distributed log is preserved: a log consists of blocks, blocks include transactions. For most votes, this means that each vote is a transaction that will subsequently be combined into blocks based on the block size or timeout and stored in a common journal. In the case of the scheme described in this paper, the block consists exclusively of one bulletin from one user, that is, there is no additional level of the blockchain hierarchy. A similar architecture is typical for Byteball Bytes DAG DLT, where the concept of a block is absent and each transaction contains information about the previous one. The system allows for exclusively public voting without hiding any details of the procedure.

The authors of the article [8] drew attention to another aspect of modern electronic voting protocols - public key infrastructure (PKI). Most existing blockchain voting systems are based on a bunch of cryptographic keys, the reliability of which is guaranteed by number theory for modern computers. However, the rapid development of quantum technologies is a threat to modern cryptographic algorithms, which can become a serious threat to existing systems using cryptographic approaches. This article pays special attention to the issue of anonymization of voters. The scheme proposed in the article combines certificateless ring signature and code-based Niederreiter anonymization approaches. Protection from quantum attacks, which has become the main feature of the described system, is provided by using the modified Niederreiter cryptosystem to distribute Partial Private Keys for each voter. The algorithm is based on code, and its security comes down to the problem of syndrome decoding coding theory. This is an NP-complete problem that is difficult to solve even for quantum computers with powerful computing power. The article provides a formal proof of algorithm security. The scheme proposed in the article is characterized by a linear relationship between security and system efficiency. The optimum value for safety and performance can only be provided for small-scale votings. An increase in the number of voters leads to an increase in the level of security of the system, having a strong influence on the system performance.

To a greater extent, performance issues were addressed in the article [9]. The article talks about blockchain voting based on the Multichain platform. Of course, blockchain is one of the breakthrough technologies of our time, used mainly in the financial sector, but the question of the possibility of comparing financial blockchain systems with traditional payment systems like VISA remains open. The article discusses the various configurations of the blockchain platform for voting in order to conduct a comparative analysis in terms of system performance and scalability. The feasibility of such an analysis becomes possible due to the great flexibility of the Multichain blockchain platform used, which allows managing not only the parameters traditional for such systems, such as block size or transaction, but also the level of access to the blockchain network (commissioned, permissionless).

3 Hyperledger Fabric-Based E-Voting System

The solution based on the Hyperledger Fabric distributed ledger that is examined for performance further is presented in [10]. Here we provide a brief description of the protocol (Fig. 1).

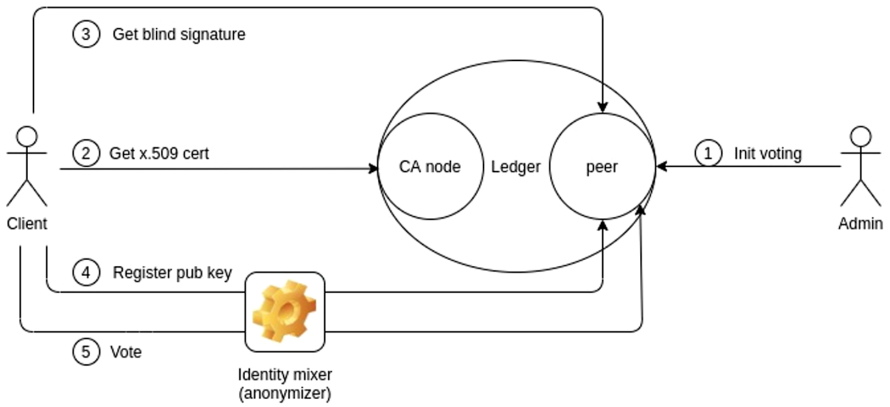

Fig. 1. High-level overview of the Hyperledger Fabric-based e-voting system

The scheme consists of the following steps:

Network Configuration. During this phase organizations, the number of nodes they have, and their rights (read, write) are determined. The network is set up, CA nodes are added, logic (chaincode) is loaded, nodes of the ordering type are determined to achieve consensus.

Voting Configuration. During this phase, the administrator generates a list of questions, determines the list of departments that can participate in this voting. Then each of the administrators of departments forms an additional data (start/end of voting and registration, list of voters) necessary for holding the voting among voters who belong to this department. Also at this stage a key is generated (which is shared between observers). This key will be used to encrypt ballots.

User Registration. This phase is crucial for several purposes: to preserve the possibility of a voter to vote using paper-based ballots (if the user has not been registered he is able to vote only in traditional way); to ensure the choice privacy and maintain eligibility.

- The sub-step of obtaining a blind signature. During the registration each user generates a pair of keys and a random number. Then the voter requests a blind signature from the chaincode. It checks that the current user is allowed to participate in this voting. If it is valid the blind signature is sent back to the user. Then chaincode registers in the ledger that the current user has received a blind signature.
- The sub-step of registering a public key that corresponds to a person is known only by the voter. At this moment the user has a signed public key which he sends it to the chaincode anonymously. Then signature is verified. If it is correct, the public key is recorded to the ledger.
- Ability to cancel registration and vote using the paper-based method. This phase is not anonymous. The voter sends to the chaincode a request to remove the registration key from the list. The chaincode does not completely delete the key but marks it as revoked. This is necessary as if the user changes his mind and wants to register again he has to register a new key. In addition, he is removed from the list of signed users.

Voting. During this phase, users who have registered their private keys for voting can send encrypted ballots to chaincode. Until the end of voting process the voter can change his decision and only the final one is counted.

Result Counting. After the voting is completed observers with private key parts upload them to the chaincode. Thus, it is possible to decrypt the ballots and to get the results that are published on the ledger.

3.1 Performance

The results of calculating the throughput of both the Hyperledger Fabric itself (in the most common configurations) and some of the test cases that were conducted showed disappointing results. For example, an article from the developers themselves [11] shows that out of the box, depending on the parameters,

the throughput can vary from 1000 to 3500 tps. Some researchers [12] show that through serious modification of some components, throughput up to 20,000 tps can be achieved. However, at the same time, researchers from Russia (Sberbank) showed that in some cases the speed will be about 1000 tps when using the built-in LevelDB database, and about 300 tps when using CouchDB.

The tests that we conducted showed intermediate results between these values. Several different network configurations were tested:

- Solo (one organization, one node, one orderer, only 5 docker containers),
- Kafka (two organizations, each with 2 nodes, 2 orderers, 3 zookeepers, 4 kafka containers, a total of 20 docker containers).

Each configuration showed similar characteristics: about 2000 tps. After that, some transactions began to fall off by timeout. Clients to interact with the ledger were developed using the following SDKs:

- Java SDK,
- Node SDK,
- Python SDK

Below are the graphs of the average execution time of the voting scenario for one user in ms (for a different number of simultaneous active users. From 100 to 1300 in steps of 200. With more users, the system stopped managing the load and either crashed or some of the users could not receive a response from systems).

The script includes:

- User registration in the HLF system (marked as reg),
- Getting available polls (each user had two polls available, one of which has already ended),
- Getting the results of the ended voting
- Registration in the ongoing vote
- Voting

Testing was carried out on various network configurations (Solo, Kafka) in HLF v1.4.1. The effect of registration was also studied (if the user does not yet have keys for interacting with the ledger, and he needs to contact the CA to get them). At the same time, everything is saved in the ledger: users, voting, bulletins, results etc.

Configuration example: “solo 2 s, 10 msg, 99 MB (reg)” means that the solo network is set up with the following block formation parameters:

- Batch Timeout: The amount of time to wait before creating a batch - 2 s
- Max Message Count: The maximum number of messages to permit in a batch - 10 msg
- Absolute Max Bytes: The absolute maximum number of bytes allowed for the serialized messages in a batch - 99 MB

– Preferred Max Bytes: The preferred maximum number of bytes allowed for the serialized messages in a batch. A message larger than the preferred max bytes will result in a batch larger than preferred max bytes - 512 KB by default

Mark "(reg)" means that users had to register first.

If the fourth parameter is present in the configuration name, then it defines "PreferredMaxBytes" for forming the block.

The plots below illustrate the comparison of operating time in the same configurations depending on the need to initially register in the system (Fig. 2). It can be seen that if it is necessary to register, the execution time of the script increases by 1,5–2 times. This is due to the fact that during registration the user receives a new X.509 certificate, the formation of which is an expensive cryptographic operation. Average voting time of performing the whole voting scenario for a single user is measured.

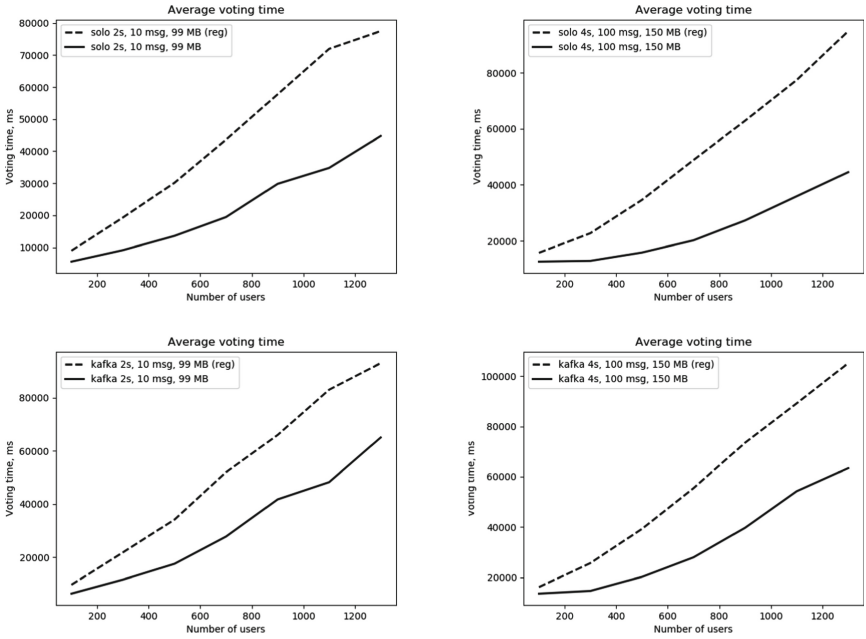

Fig. 2. Comparison of average voting time depending on the need of registration

Comparison of the operating times of different types of consensus (solo and kafka) in the same configurations is presented in Fig. 3. The execution time of the script in the case of Solo was expectedly shorter due to the fact that there is actually only one node that orders transactions, which allows us not to spend resources on consensus between nodes as it happens in Kafka.

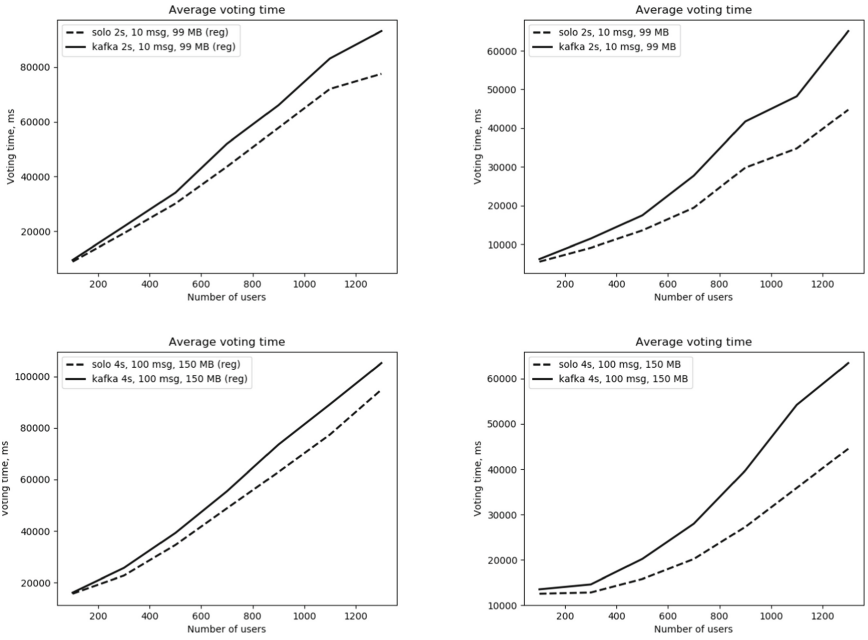

Fig. 3. Comparison of average voting time depending on the type of consensus

Comparison of the operating time with the same type of consensus, but with different parameters of the formation of the block is shown in Fig. 4. It can be seen that an increase in block size when registration was required did not give a performance gain. This is due to the fact that the registration itself takes a comparable time in time to complete the rest of the script. In the case of its absence in the Kafka consensus, we see an increase, especially noticeable after 900 simultaneously active users.

4 Theoretical Alternatives

Identity Mixer (Idemix) technology was developed by IBM and is based on cryptographic techniques, the development of which was presented at the conference and carefully tested by the community. The practical implementation is based on the scheme, which can be found in more detail in [13–15]. Here we provide only a brief description of this technology.

Suppose that there are three participants: the user, issuer (the authority issuing the certificate and confirming the identity of the user) and verifier, which wants to make sure that the user really has some attributes that are necessary to perform some action. In such a scheme, the main problem that Idemix solves is providing to the verifier not all the information about the user, but only certain attributes, or even just proving that the requested attribute falls into a certain range. In addition, it is possible to provide unlinkability, that is, the inability to

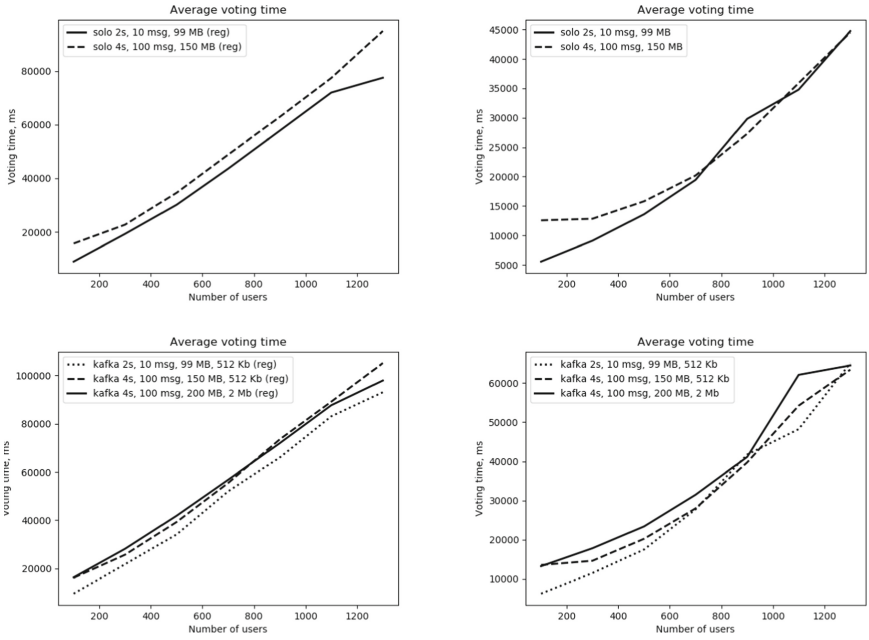

Fig. 4. Comparison of average voting time depending on the block formation parameters

connect two different requests with the same user. In this way, Idemix allows for anonymity and privacy.

Let us briefly describe how Identity Mixer works. Some users want to get Idemix credentials, for this they need to interact with CA:

1. Initialization of receiving credentials.
2. CA returns nonce starting the procedure.
3. The user generates a secret (random number) and on the basis of it, as well as using the public key CA and nonce from the previous step, forms a request, which consists of a commitment's secret and evidence with zero disclosure that the user really knows this secret.
4. CA verifies the evidence and puts its signature on this data along with additional attributes (for example, whether the user belongs to an organization). At the end, this data is returned to the user.
5. The user verifies the CA signature and saves credentials.
6. Now the user can use these credentials to sign messages when communicating with someone. The examiner will be able to make sure that the user really has a signature from the CA, while he will additionally only know the information that the user wants to share.

4.1 Architecture

Registration in the System. To interact with the system, the user must be registered in it. This can be done either simply using a username and password, or using an X.509 certificate.

Create a Vote. When creating a voting, the administrator must specify a list of users who are already registered in the system and who need to access the voting.

Confirmation of Participation in the Vote. After that, users must confirm their participation in this vote. This is achieved with an identity mixer. That is, they make a request for new Idemix credentials, in which the voting identifier acts as an additional attribute on which the signature is placed from the CA.

Voting. Using the credentials received earlier, users send anonymous bulletins (which is one of the main requirements for any voting system). Moreover, verification that this user really has the right to vote is checked on the basis of the signed voting identifier. Users can send their vote several times, only the last one will be taken into account. This makes senseless to enforce to vote in the presence of an evil observer behind.

4.2 Comparison

Let us highlight the main similarities and differences in the two approaches and try to understand whether the absence of a blockchain is critical.

Lack of Additional Cryptographic Operations. In the previous approach, an RSA-based blind signature was used to register participants in each voting. Then, with anonymous access to the ledger, Idemix credentials were used. Now the first part is missing, or rather, is integrated into obtaining Idemix credentials (since the release of this certificate already uses blind signature BSS+).

Mobile Client Support. For the previous solution, there is only one official SDK that supports Idemix, for the Java language. This greatly complicates the development of native client applications for iOS. When using the new solution, it is possible to:

- Writing all-Go native mobile applications,
- Writing SDK applications by generating bindings from a Go package and invoke them from Java (on Android) and Objective-C (on iOS).

Support for Voting Properties. Despite the blockchain dismissal, the security features that were inherent in the previous solution were preserved:

- Eligibility - the user gets access to the vote only when the creator of the vote added it to the list. This is checked when the user confirms the participation in the vote, when he tries to get idemix credential. If he is on the list, he will receive a signed attribute that contains a voting identifier; if not, he will be refused to issue new credentials.
- Un-reusability - when issuing idemix credentials, CA puts a blind signature on the public key, which will be used to determine that the request came from the same anonymous person to exclude double counting
- Un-traceability - it will not work to track the user, since the key under which the ballots are stored in the database is known only to the user (when registering in a vote, he signs with CA blind signature, that is, CA does not know which key is signed), and all interactions are Voting times with the system occur using an identity mixer, that is, anonymously.
- Verifiability - since the user knows his key, he can make sure that in the database in conjunction with this key is exactly the newsletter that he sent.

Absence of Blockchain. The main advantage that a distributed ledger can provide is the lack of centralization and the inability to unilaterally change the data in the database. For state elections this is really important, for corporate voting this might be required to a lesser extent. But let us show what pitfalls may appear. The previous solution allowed connecting external observers by connecting new nodes to the network. However, such observers could not be arbitrarily many (due to scaling problems) and only large organizations became them. In addition, the user alone cannot initially access the network. This happens through authentication in the organization to which this user belongs. And it is the organization (a higher level of hierarchy) that owns the nodes in the network that can take part in reaching consensus. This is a limitation of the Hyperledger Fabric ledger, which is classified as permissioned. Using public blockchains is not suitable again, due to scalability issues, when we have hundreds of thousands or even millions of users, the throughput becomes so low that users cannot send their ballots because the network is congested. That is, the final voter is forced to trust his organization, which issues his identity and within which he joins the distributed network. Thus, one of the main properties of the blockchain is not fully implemented, if not absent at all. It can be argued that the organization cannot change the user's voice, as the transactions are confirmed by other parties. However, a certificate may be revoked or reissued from the user, or the organization may spawn fictitious users in order to achieve the desired voting result.

5 Conclusion

This article conducted a study of the performance of an electronic voting system using distributed ledger technologies. It is shown that in some cases it is advisable

to abandon the use of the blockchain without losing the security properties. A theoretical scheme is proposed that can allow anonymous voting using Identity Mixer technology. The proposed solution suggests the possibility of developing native client applications for iOS and Android.

In order to improve the performance of the system, it is proposed to add an additional module that will help ensure receipt-freeness; this can be done using, for example, the approach from [16]. This is the direction of our future work.

References

1. IP Chain association. https://www.ipchain.ru/association/
2. NEO: Open Network for the Smart Economy. https://neo.org/
3. EOS.IO Technical White Paper v2. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md
4. Georgiadis, E.: How many transactions per second can bitcoin really handle? Theoretically. https://eprint.iacr.org/2019/416.pdf
5. A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper
6. Menon, A., Bhagat, V.: Blockchain based e-voting system. Indian J. Appl. Res. **10**(1), 8–10 (2020)
7. Yi, H.: Securing e-voting based on blockchain in P2P network. EURASIP J. Wirel. Commun. Netw. **2019**(1), 1–9 (2019). https://doi.org/10.1186/s13638-019-1473-6
8. Shiyao, G., Dong, Z., Rui, G., Chunming, J., Chencheng, H.: An anti-quantum e-voting protocol in blockchain with audit function. IEEE Access **7**, 115304–115316 (2019). https://doi.org/10.1109/ACCESS.2019.2935895
9. Khan, K.M., Arshad, J., Khan, M.M.: Investigating performance constraints for blockchain based secure e-voting system. Future Gener. Comput. Syst. **105**, 13–26. ISSN 0167–739X. https://doi.org/10.1016/j.future.2019.11.005
10. Kirillov, D., Korkhov, V., Petrunin, V., Makarov, M., Khamitov, I.M., Dostov, V.: Implementation of an e-voting scheme using hyperledger fabric permissioned blockchain. In: ICCSA 2019. Lecture Notes in Computer Science, vol. 11620. Springer, Cham (2019)
11. Androulaki, E., et al.: Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference (EuroSys '18), Article 30, pp. 1–15. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3190508.3190538
12. Christian, G., Stephen, L., Lukasz, G., Srinivasan, K.: FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second, pp. 455–463 (2019). https://doi.org/10.1109/BLOC.2019.8751452
13. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_4
14. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) Security and cryptography for networks. In: SCN 2006, LNCS, vol. 4116. Springer, Berlin, Heidelberg (2006)
15. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong Diffie Hellman assumption revisited. Cryptology ePrint Archive, IACR (2016)
16. Shinsuke, T., Hazim, H., Nazmul, I., Kazi, A.: An incoercible e-voting scheme based on revised simplified verifiable re-encryption mix-nets. Inf. Secur. Comput. Fraud **3**, 32–38 (2015). https://doi.org/10.12691/iscf-3-2-2