

The Combination of P-BFT and RAFT: A New Approach to Building Networks that Provide Reliability and Security

Alexander Bogdanov[1,2], Nadezhda Shchegoleva[1,2], Valery Khvatov[3], Gennady Dik[1,4], Jasur Kiyamov[1(✉)], and Aleksandr Dik[1]

[1] St. Petersburg University, Saint Petersburg, Russia
{a.v.bogdanov,n.shchegoleva,z.kiyamov,a.dik}@spbu.ru
[2] St. Petersburg State Marine Technical University, Saint Petersburg, Russia
[3] DGT Technologies AG., Toronto, Canada
[4] St. Petersburg LLC "System Technologies", Saint Petersburg, Russia
g.dick@systechnologies.ru

Abstract. The article considers two data processing algorithms in distributed systems: P-BFT and Raft. Both algorithms are designed to ensure data consistency and avoid conflicts in distributed environments. However, they have different approaches to solving these problems. The P-BFT algorithm operates at the bottom layer of data access and uses the principles of distributed consensus to achieve high transaction processing speed. Raft, in turn, operates at the top layer of data access and provides guarantees for data integrity and system state consistency. The article discusses the advantages and disadvantages of both algorithms and the choice between them depends on the requirements and characteristics of a particular system. Recommendations are given on the use of hybrid algorithms for more efficient operation of the system.

Keywords: blockchain · distributed system · RAFT · P-BFT

1 Introduction

In the modern world, blockchain technologies are increasingly used in various fields of activity, including finance, healthcare, government services and much more. One of the key aspects of blockchain technologies is the consensus algorithm, which allows reaching agreement between different network participants. One of the most common consensus protocols is the RAFT protocol, which was designed to provide simplicity and clarity in the process of leader selection and decision making in distributed systems. However, RAFT has some disadvantages that may limit its use in more complex systems.

Another common consensus protocol is P-BFT, which provides high speed and reliability in the process of negotiation between network participants. However, P-BFT also has disadvantages associated with high complexity and computational costs. To overcome the shortcomings of both consensus protocols, a layered network was developed based on a combination of the P-BFT and RAFT

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14104, pp. 572–583, 2023.
https://doi.org/10.1007/978-3-031-37105-9_38

protocols. This network allows for higher speed and reliability in the negotiation process between network participants, and also provides support for security certificates to ensure security in the leader election and decision-making process.

A multi-layer network based on a combination of P-BFT and RAFT protocols consists of two layers: the first layer uses the P-BFT protocol to negotiate between nodes, and the second layer uses the RAFT protocol to select a leader and make decisions if a quorum was not reached at the first layer. At the same time, each node has a security certificate that ensures the authenticity and integrity of information during the negotiation process.

The first layer of a layered network based on the P-BFT protocol works as follows: each node sends a request to add a new block to the network's block chain, and other nodes must confirm it. At the same time, in order to reach consensus, it is required that the majority of nodes confirm the request. If an acknowledgment has not been received within a certain time, the request is sent to the second layer of the network based on the RAFT protocol.

At the second level of the network, based on the RAFT protocol, a leader is selected who makes a decision on behalf of the entire network. This happens with the help of elections that are held between the nodes of the network. The leader receives requests that were not confirmed at the first level, and makes a decision based on them. At the same time, each node has the ability to verify the authenticity and integrity of the solution using a security certificate.

Thus, a multilayer network based on a combination of P-BFT and RAFT protocols provides high speed and reliability in the negotiation process between network participants, as well as security support in leader selection and decision making. This makes it one of the most promising models for use in various fields of activity where fast and reliable processing of a large number of transactions is required while ensuring the security and authenticity of data.

2 Data Processing at the Lower Access Level

The architecture of the combined approach of P-BFT and RAFT consensus includes the use of P-BFT as the lower layer protocol for fast and reliable transaction processing and RAFT as the upper layer protocol for guaranteed delivery and state negotiation between nodes in the network.

At the lower level, the P-BFT algorithm is used, which provides fast and reliable transaction matching in real time. Network nodes are divided into groups of K nodes, where K is the number of nodes required to confirm the transaction. If a node receives a transaction, it sends it to other nodes in its group. When K nodes confirm a transaction, it is considered confirmed.

At the top level, the RAFT algorithm is applied, which provides guaranteed delivery and state negotiation between nodes in the network. The nodes of the network choose one node as the leader, which is responsible for distributing state among the nodes. If a node wants to update its state, it sends a change request to the leader. The leader receives the request, processes it, and sends a response with the new state. When a majority of nodes acknowledge a state change, it is considered to be committed.

The combined approach of P-BFT and RAFT provides fast and reliable transaction processing at the lower layer and guaranteed delivery and state negotiation at the upper layer. This makes it more fault tolerant and more efficient in handling a large number of transactions.

For data processing at the lower P-BFT data access layer, consider in more detail:

1. Develop a data transfer protocol that will be used by nodes to exchange messages and transactions in the system.
2. Implement a mechanism for authenticating messages and transactions using cryptographic methods such as message signing and digital certificates.
3. Implement low level data access signature and authentication algorithms to detect and reject messages and transactions that may be forged or unauthorized.
4. Use P-BFT distributed consensus algorithms to determine the order in which transactions are added to the blockchain and resolve possible conflicts when creating new blocks.
5. Implement mechanisms for collecting, storing and processing data at the lower level of P-BFT data access, such as block storage, data integrity checking, and synchronization between nodes.

Data processing at the lower layer of P-BFT data access can be quite complex and resource-intensive [1,2]. However, with proper design and optimization of algorithms, it is possible to achieve high performance and reliability of a distributed system.

Data processing at the lower layer of P-BFT data access includes the development of protocols (Fig. 1) [5], authentication and authentication mechanisms, consensus algorithms, data storage and processing. All of these steps are important to create a reliable and secure distributed system based on P-BFT [3,4]. It is also important to optimize algorithms and system design in order to achieve high performance and scalability. The following steps can be used to develop a communication protocol in a P-BFT system:

1. Define the format of messages and transactions. It is necessary to determine what data will be transferred between nodes and how it will be packaged into messages and transactions.
2. Define the protocol for establishing a connection between nodes. It is necessary to determine how the nodes will find each other and establish a connection for the transmission of messages and transactions.
3. Develop mechanisms for checking the integrity of messages and transactions. This may be implemented using digital signatures and hash functions.
4. Develop mechanisms for error handling and recovery after failures. It is necessary to determine how the system will handle errors such as message loss, message and transaction retransmission, and system recovery from failures.
5. Define a consensus protocol to determine the order in which transactions are added to the blockchain. This can be implemented using the P-BFT consensus algorithm.

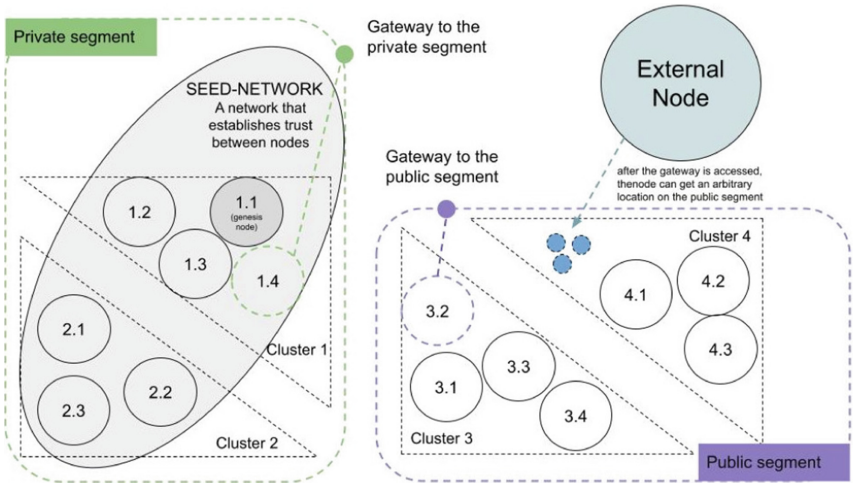


Fig. 1. Network Topology and Node Attaching

6. Develop synchronization mechanisms between nodes to ensure a single state of the system. This can be implemented using blockchain synchronization mechanisms between nodes.
7. Develop mechanisms to protect against attacks. It is necessary to provide mechanisms to protect against various types of attacks, such as denial of service (DDoS) attacks, consensus attacks, and others.

It is important to make sure that the developed data transfer protocol provides high performance, scalability and reliability of the P-BFT system [6]. To do this, it is necessary to test and optimize the data transfer protocol.

The implementation of the message and transaction authentication mechanism is one of the important steps in the development of a data transfer protocol in a distributed system, especially in open access systems where any node can send messages and transactions. Cryptographic methods can ensure the security of data transmission, protect them from changes and forgeries.

One way to secure data transmission in a distributed system is to use message signatures and digital certificates. The message signature allows you to verify that the sender of the message is really who he claims to be, and the digital certificate is a kind of seal that confirms the authenticity of the public key of the sender of the message [7]. Implementing a message and transaction authentication mechanism using message signing and digital certificates typically involves the following steps:

1. Generation of a key pair (public and private) for each node in the system.
2. Signing messages and transactions by a node using its private key.
3. Verification of the signature of messages and transactions by other nodes using the public key of the sender.

4. Obtaining a digital certificate to verify the public key of the sender of a message or transaction.
5. Checking a digital certificate for authenticity and validity.
6. Checking the sender's public key against its digital certificate.

The implementation of the message and transaction authentication mechanism is an important step in the development of a data transfer protocol in a distributed system and should be taken into account when designing the protocol [8–10]. The following algorithms can be used to verify signatures and authentication at a low data access level:

1. EDS algorithm (Electronic Signature Algorithm) - used to create and verify digital signatures of messages and transactions. The private key is used to create the signature, and the public key is used to verify the signature.
2. Hash Algorithms - used to create hash values of messages and transactions. A hash value is a unique number that can be used to verify the integrity of a message or transaction.
3. Symmetric and Asymmetric Encryption Algorithms are used to protect the confidentiality of messages and transactions. Symmetric encryption uses the same key to encrypt and decrypt messages, while asymmetric encryption uses a public/private key pair.
4. Digital Timestamping Algorithms - used to protect against re-sending of messages and transactions. A timestamp is a unique value that indicates when a message or transaction was sent, and which is also digitally signed.

When implementing algorithms for verifying signatures and authentication, it is necessary to take into account the requirements for the security and performance of the system, as well as possible attacks, such as spoofing messages and transactions, attacks on hash functions, etc.

The algorithm allows the nodes of the system to reach agreement on the order in which transactions are added to the blockchain and the resolution of possible conflicts when creating new blocks. The P-BFT algorithm uses several steps to reach consensus:

1. Preliminary stage - the nodes send a request message to perform an operation to one of the leaders of the system.
2. Registration stage - the leader receives the request and distributes it among the system participants, who verify the message signature and its authenticity.
3. Pre-commit stage - participants send a message confirming the signature and pre-commit of the operation.
4. Commit stage - after receiving confirmations from all participants, the leader creates a new block in the blockchain and sends a message to all nodes of the system that the operation is completed.
5. Thus, the P-BFT algorithm allows reaching a consensus in the system and ensuring the reliability and security of transaction processing.

To use the P-BFT algorithm in a data processing system at the lower access level, it is necessary to implement its algorithmic logic and ensure interaction between the system nodes for messaging and performing operations [10,11]. The following steps can be used to implement mechanisms for collecting, storing and processing data at the lower level of P-BFT data access:

1. Block storage: At the bottom layer of P-BFT data access, it is necessary to implement a block storage mechanism that contains transactions and other information about the state of the system. This can be done using a database or other data storage mechanisms such as distributed file systems.
2. Data Integrity Check: To ensure the reliability of the system, it is necessary to check the integrity of data, including blocks, transactions, and other elements. To do this, you can use cryptographic methods such as hashing and digital signatures.
3. Synchronization between nodes: to ensure data consistency between the nodes of the system, it is necessary to implement synchronization mechanisms. This may include mechanisms for propagating blocks and transactions between nodes, as well as mechanisms for detecting and resolving conflicts.
4. Data processing: At the bottom layer of P-BFT data access, it is necessary to implement data processing mechanisms, including checking access rights, executing transactions, and other operations.
5. Monitoring and control: In order to effectively manage the system, it is necessary to implement monitoring and control mechanisms. This may include mechanisms for monitoring the state of the system, managing access rights, and other functions.

The implementation of these mechanisms can be complex and resource intensive, so careful system design and optimization is required.

3 Top Level Data Processing

RAFT consensus is a distributed consensus algorithm that is used to reach agreement in multi-node systems. It provides leadership, which is responsible for making decisions and distributing tasks between nodes [1,12]. The top level of data processing in RAFT consensus includes the following steps:

1. Formation of requests. Each node generates requests to perform operations and sends them to the leader.
2. Leader acknowledgment. The leader receives the requests and responds to each one with an acknowledgment. The acknowledgment contains information that the request has been received and accepted for execution.
3. Replication leader. The leader replicates requests to other nodes in the system.
4. Replication confirmation. Nodes that receive requests respond with a replication acknowledgment. This ensures that all nodes received the request and accepted it for execution.

5. Application of operations. The nodes perform the operations specified in the requests and acknowledge their successful completion to the leader.
6. Journal update. The leader updates the log containing all completed operations and sends it to the nodes.
7. Update confirmation. The nodes acknowledge receipt and application of the updated log.
8. Continued work. After the successful completion of all stages, the nodes continue their work and wait for new requests.

Some implementations of RAFT consensus also include mechanisms for detecting and recovering from system failures to ensure reliable operation even when errors occur.

In the Raft protocol, each node can generate requests to perform operations and send them to the leader. The leader receives requests from all nodes and logs them in its operations log. The leader then replicates the log to the remaining nodes using the replication mechanisms described in the protocol.

Each request has a unique identifier and may contain data that needs to be processed. The leader chooses the order in which operations are performed based on a logical time called a term. If the leader changes, then the new leader can continue executing requests that were not fulfilled by the previous leader.

When forming requests, it is important to consider that some operations can change the state of the system, while other operations can only receive information. In the case of a change in the state of the system, it is necessary to ensure the atomicity of the execution of operations in order to avoid an inconsistent state. To do this, the Raft protocol uses a mechanism for committing operations, which ensures that the operation will be executed only after it has been replicated to the majority of nodes.

Leader confirmation is one of the steps in the Raft algorithm to achieve consensus among nodes in a distributed system. Once the nodes have agreed on which node will become the leader, that leader begins to receive requests to perform operations from other nodes [12,15].

Each node, sending a request to the leader, waits for confirmation that its request has been received and accepted for execution. The leader in response to each request sends an acknowledgment that contains information that the request has been received and will be fulfilled. This acknowledgment is known as a “request response” or “request-response”.

After receiving confirmation, the sender of the request goes into the “waiting for a response” state and waits for the leader to complete the operation and inform all nodes in the system about it [13,14].

If the leader is unable to complete the request due to some error, then it will send an error message to the node that sent the request instead of an acknowledgment. Errors can occur, for example, if the requested resource is not available or if the leader has lost contact with most of the nodes in the system. After receiving an acknowledgment, the sending node can continue to work, knowing that its request was successfully accepted and completed.

4 Combined Access Processing Algorithm

Data processing algorithm at the lower level of P-BFT data access (Fig. 2):

- Step 1: Each node generates transactions based on requests it receives from users.
- Step 2: Nodes send their transactions to other nodes in the system.
- Step 3: Each node verifies the authenticity and integrity of transactions using cryptographic methods such as digital signatures and certificates.
- Step 4: Nodes use P-BFT distributed consensus algorithms to determine the order in which transactions are added to the blockchain and resolve possible conflicts when creating new blocks.
- Step 5: Nodes synchronize their blockchains to ensure data integrity and homogeneity.

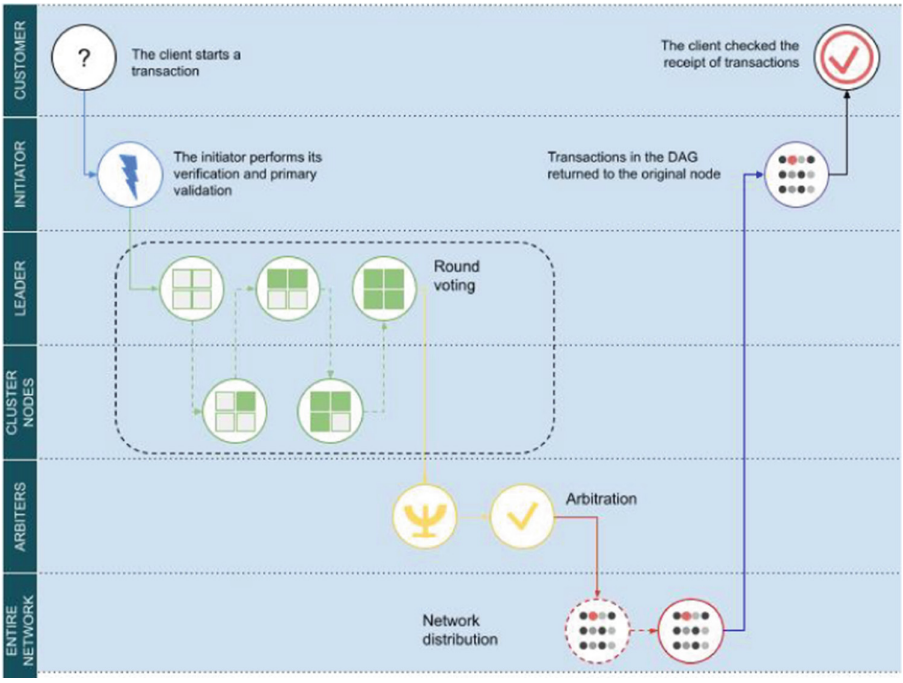


Fig. 2. Data processing at the lower data access layer P-BFT

Algorithm for processing data of the upper level of RAFT data access:

- Step 1: Each node generates requests to perform operations and sends them to the leader.

– Step 2: The leader receives the requests and responds to each with an acknowledgment.
– Step 3: When the leader receives confirmation from the majority of the participants, he performs the operations and sends the results back to the participants.
– Step 4: Participants accept the results, verify their authenticity and integrity, and update their local copies of the data.
– Step 5: If the leader fails, then the participants choose a new leader from their ranks.

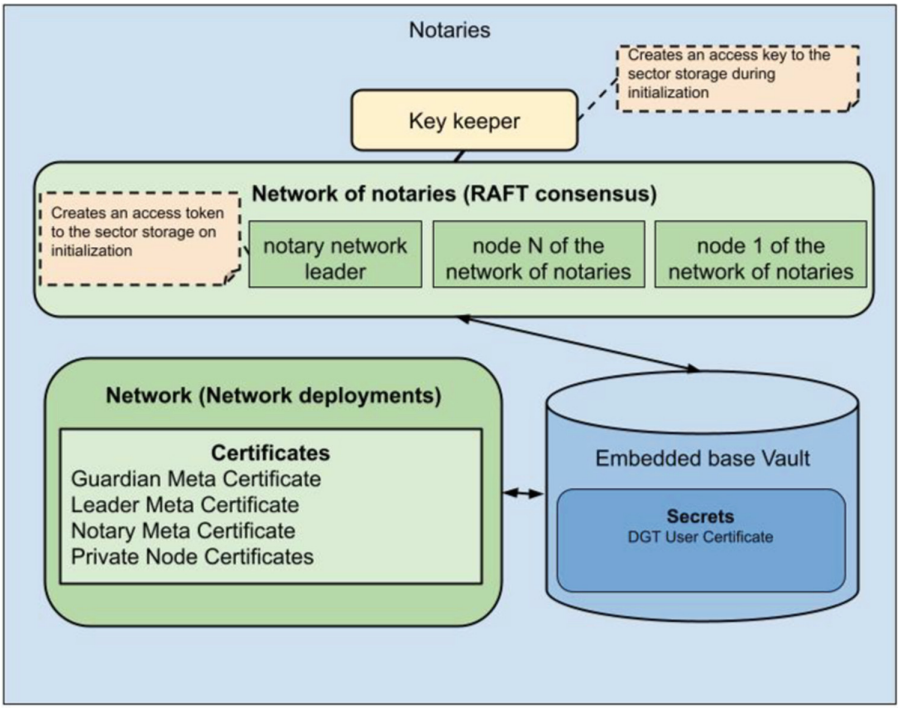


Fig. 3. Data processing at the lower data access layer P-BFT

The numerical method of a multilayer network based on a combination of the P-BFT and RAFT protocols can be implemented using algorithms that provide high transaction processing speed and data reliability in a distributed environment. It takes the following steps:

1. Creation of a multilayer network based on a combination of P-BFT and RAFT protocols with several nodes and levels;
2. Separation of nodes into groups that can work independently of each other and process transactions in parallel;

3. Implementation of a consensus mechanism based on the P-BFT and RAFT protocols, which allows achieving unity among nodes and maintaining data reliability;
4. Optimization of the transaction processing process, for example, by using parallel computing and load distribution between nodes;
5. Implementation of protection mechanisms against various types of attacks, including DoS attacks and "transaction manipulation" attacks, to ensure the security and reliability of the network;
6. Implementation of mechanisms for monitoring and managing the network, including tools for analyzing and monitoring performance, mechanisms for detecting and correcting network errors and problems, as well as mechanisms for updating and scaling the network;
7. Ensuring that the network is sufficiently scalable and flexible to adapt to changing user needs and changing conditions in a distributed environment;
8. Implementation of mechanisms for storing and managing data in the network, including mechanisms for backing up, restoring and synchronizing data between nodes;
9. Development and implementation of interfaces and applications that allow users to interact with the network and use it to exchange data and conduct transactions;
10. Testing and optimizing the network, including testing for strength, performance, reliability and security, as well as finding and eliminating errors and vulnerabilities;

Both algorithms have their advantages and disadvantages, and the choice between them depends on the requirements and characteristics of a particular system. P-BFT is more suitable for systems with high transaction processing speed and low error rate, while Raft is more suitable for systems where data integrity and system state consistency are important.

P-BFT and Raft have their own advantages and disadvantages, which may make them more suitable for different types of systems. For example, P-BFT may be preferred for high transaction volume systems, such as financial or banking systems, where fast processing and reliable confirmation of transactions is required. At the same time, Raft may be preferred for systems with more complex state and data integrity requirements, such as database management systems.

Also, Raft (Fig. 3) provides ease of understanding and implementation, while P-BFT requires more complex implementation and can be more difficult to understand. It is also important to note that both algorithms can be expensive to use due to the extra cost of messaging between system nodes to maintain consensus. Thus, the choice between P-BFT and Raft should be based on a balance between speed requirements and data integrity and system state consistency requirements.

An important aspect of a multilayer network based on a combination of P-BFT and RAFT protocols is the support for security certificates. Security certificates are used to provide security in the leader selection and decision making process. Security certificates can be used to authenticate network partici-

pants, thereby providing protection against possible message spoofing or spoofing attacks. In addition, certificates can be used to secure the leader election process by ensuring that only trusted participants can participate in the leader election process.

In a layered network based on a combination of the P-BFT and RAFT protocols, certificates can be used to authenticate the members of each layer of the network, as well as to secure the leader election and decision making process at each layer. In addition, certificates can be used to ensure the confidentiality and integrity of data transferred between network participants.

Support for security certificates is an important aspect of a multi-layer network based on a combination of P-BFT and RAFT protocols, ensuring the security of the leader election and decision-making process at each layer of the network.

5 Conclusions

In conclusion, we can say that both P-BFT and RAFT algorithms are effective tools for data processing and ensuring reliable and secure operation of distributed systems. P-BFT is suitable for systems with high transaction processing speed and low error rate, while RAFT is more suitable for systems where data integrity and system state consistency are important. The choice between them depends on the requirements and characteristics of a particular system, so it is recommended to conduct a thorough study and comparison of these algorithms before applying them to specific projects.

In general, the choice of an algorithm for processing data in a distributed system should be based on a comprehensive analysis of the requirements and characteristics of the system, as well as on the study of possible options for solving problems related to the reliability and safety of the system.

It seems to us that only a hybrid approach, i.e. a combination of both algorithms can provide the required level of security and transaction speed.

The article was prepared as part of the implementation of the program of strategic academic leadership “Priority-2030”, Strategic project No. 2 “Digital transformation of the university and industrial enterprises based on the Intelligent Cyber-Physical Platform”.

References

1. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation, pp. 173–186 (1999)
2. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, 2nd edn. CRC Press, Boca Raton (2014)
3. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008).https://bitcoin.org/bitcoin.pdf
4. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)

5. Bogdanov, A., et al.: A multilayer approach to the security of blockchain networks of the future. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) Computational Science and Its Applications - ICCSA 2022 Workshops, ICCSA 2022. Lecture Notes in Computer Science, vol. 13377, pp. 205–216. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10536-4_14
6. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: IEEE International Conference on Blockchain, pp. 112–121 (2015)
7. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: IEEE International Congress on Big Data, pp. 557–564 (2017)
8. Guerraoui, R., Kouznetsov, P., Vukolić, M.: The Paxos-Killer: fault-tolerant Byzantine consensus using linked transactions. Distrib. Comput. **32**(2), 107–125 (2019)
9. Miller, A., Juels, A., Shi, E., Parno, B.: The perils of zero knowledge protocols for blockchains. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 905–917 (2017)
10. Dolev, D., Fichman, V., Lotker, Z.: The making of a fastest Byzantine fault-tolerant consensus algorithm. J. Cryptol. **30**(1), 242–270 (2017)
11. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 120–130 (1999)
12. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_9
13. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: USENIX Annual Technical Conference, pp. 305–320 (2014)
14. Howard, H.G., Malkhi, D.: A brief survey of practical Byzantine fault tolerance. ACM SIGACT News **47**(2), 51–59 (2016)
15. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. **16**(2), 133–169 (1998)