Information Sciences 559 (2021) 8–21

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

DRBFT: Delegated randomization Byzantine fault tolerance consensus protocol for blockchains

Yu Zhan [a,b], Baocang Wang [a,b,*], Rongxing Lu [c], Yong Yu [d]

[a] The State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[b] Cryptographic Research Center, Xidian University, Xi'an 710071, China
[c] The Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
[d] The School of Computer Science, Shaanxi Normal University, Xi'an 710062, China

ARTICLE INFO

Article history:
Received 12 May 2020
Received in revised form 17 October 2020
Accepted 27 December 2020
Available online 26 January 2021

Keywords:
Blockchain
Consensus protocol
Randomicity
Byzantine fault

ABSTRACT

Blockchain, as a potentially revolutionary technology, has been used in cryptocurrency to record transactions chronologically among multiple parties. Due to the fast development of the blockchain and its de-centralization, blockchain technology has been applied in broader scenarios, such as smart factories, supply chains, and smart cities. Consensus protocol plays a vital role in the blockchain, which addresses the issue of reaching consensus on transaction results among involved participants. Nevertheless, with the complexity of the network environment and growing amount of network users, the advance of blockchain is gradually restricted by the efficiency, security and reliability of consensus protocols. In this paper, we propose a delegated randomization Byzantine fault tolerance consensus protocol named DRBFT based on Practical Byzantine Fault Tolerance (PBFT) to enhance the efficiency and reliability of the consensus procedure. Specifically, a random selection algorithm called RS is developed to cooperate with the voting mechanism, which can effectively reduce the number of nodes participating in the consensus process. Our proposed scheme is characterized by the unpredictability, randomicity and impartiality, which accelerate the system to reach consensus on the premise of ensuring the system activity. Furthermore, the feasibility of our proposed scheme is also proved by both theoretical analysis and experimental evaluations.

1. Introduction

A blockchain is a time-ordered distributed ledger used to record transactions, and is maintained by multiple nodes through peer-to-peer networks [1]. Since Nakamoto proposed the first cryptocurrency–BitCoin in 2008 [2], the emerging financial system model has gradually attracted public attention. As a result, various cryptocurrencies have been proposed, such as LiteCoin [3], PPCoin [4], and BlackCoin [5]. As the basic technology of cryptocurrencies, blockchain has been closely followed by researchers due to its de-centralization and the feature that data could not be tampered with. In blockchain systems, each newly generated block is appended with the latest transactions. Then, the block is linked to the previous one after all nodes verifying its validity. Only transactions attached to the block will be executed in practice. Notice that there is no authority throughout the process since blockchains can provide trust by themselves. In addition, the security and efficiency

* Corresponding author.
E-mail addresses: yzhan1993@163.com (Y. Zhan), bcwang@xidian.edu.cn (B. Wang), rlu1@unb.ca (R. Lu), yuyong@snnu.edu.cn (Y. Yu).

https://doi.org/10.1016/j.ins.2020.12.077
0020-0255/

of blockchain systems are also ensured by cryptographic protocols. With the deepening of research, blockchains get rid of the restriction of cryptocurrencies and provide trust services for more applications in the smart contract [6,7], smart city [8] as well as medical information management [9] etc.

According to the degree of de-centralization, blockchains can be divided into three types [10], public chain, consortium chain and private chain. The characteristic of the public chain is that arbitrary nodes are allowed to participate in the process of consensus anytime. Therefore, this kind of chain is usually used in scenarios where privacy concerns and trusted institutions are not required, such as Bitcoin [2] and Ethereum [11]. The consortium chain is managed by several consortium or institutions, which means that the consortium chain is partially decentralized and supervised. For the private chain, nodes are controlled by an institution, and only authorized nodes can participate in various affairs.

The consensus protocol is one of the most critical technologies in the blockchain [12]. Due to the high latency in the peer-to-peer network, the order of transactions observed by each node is inconsistent. To solve this problem, consensus protocols were proposed to enable all nodes to reach a consensus on the content and order of transactions that created within a certain time. Different consensus protocols are suitable for different application scenarios according to their characteristics. For the public chain, there are several consensus protocols that have been successfully applied, such as Proof-of-Work (PoW) [2], Proof-of-Stake(PoS) [4] and Delegated Proof-of-Stake (DPoS) [13]. As an extensive consensus protocol, PoW requires each node to compete for accounting rights through calculating a difficult problem. This protocol accesses the consensus issue among different nodes, but leads to serious waste of resources and the lack of fairness due to the concentration of global computational power [14]. The rest protocols mentioned above also face with the potential risks which are triggered by intentional or unintentional misoperations, network delay, collapse, and system vulnerabilities.

As the size of blockchain nodes expands, the hazards posed by these risks are so serious that they cannot be ignored. Consequently, fault-tolerant consensus protocols are becoming increasingly important. Practical Byzantine Fault Tolerance (PBFT) [15] proposed by Castro et al. is the first practical consensus protocol which could tolerant Byzantine faults in an asynchronous network environment. PBFT system consists of a primary, $(n-1)$ backups and several clients. After five stages for peer-to-peer message interactions, the primary and backups reach a consensus on the request of client. In the whole process, PBFT guarantees the liveness and security of the system when the number of abnormal nodes is less than $(n-1)/3$. In addition, the consistency and correctness of consensus of PBFT are better than other protocols. However, when the amount of the participating nodes is large, PBFT is inefficient since the complexity of PBFT is high. Hence, it is a research hotspot to design PBFT based consensus protocols, which are suitable for the scenarios that have a large number of nodes and need to tolerate Byzantine faults.

Although there are some such protocols, e.g. Delegated Byzantine Fault Tolerance (DBFT) [16] and the consensus protocol used in Thunderchain [17], they lack performance analyses or formal security proofs. Therefore, the claimed performances of these consensus protocols are worrisome and unconvincing. More importantly, a lack of fairness idea of existing protocols make it difficult for system to maintain activity. In other words, an unfair systemic mechanism will reduce the enthusiasm of node participation, leading to the blockchain system losing its advantages and attractiveness.

In this paper, we propose a novel consensus protocol called delegated randomization Byzantine fault tolerance (DRBFT) to solve the problem of efficiency and activity in the prior art. The proposed protocol is suitable for blockchains with large-scale nodes in the asynchronous network environment, and reaches consensus efficiently in the case of considering Byzantine faults. The main idea of our scheme is to select delegate nodes fairly from all participating nodes to execute PBFT. In order to achieve this goal, we present a random selection (RS) algorithm that selects a specified number of nodes from a given set of nodes. Specifically, our proposed scheme is divided into three steps. Initially, a fixed number of candidates are selected from all nodes of the blockchain by voting. Each node votes for its preferred node in a period only once. After voting, the N nodes with the highest number of votes participate as candidates in the next step. Then, some candidates are selected as councillors with the RS algorithm. Finally, these councillors execute PBFT consensus protocol to generate new blocks.

In particular, the proposed RS algorithm satisfies the following properties.

- *Unpredictability*: Given a set of nodes, the results of selecting are unpredictable before executing the RS algorithm. Due to the high cost, it is impossible to achieve prejudging which nodes will be selected and interfering with the fairness of the selection.
- *Uniform distribution*: For two identical sets of nodes, the result set of running the RS algorithm is uniformly distributed and irreversible under different dynamic parameters. This property is the core of the RS algorithm. The uniform distribution of the result set means that each node is selected with the same probability. In this way, the selection can be considered fair.
- *Impartiality*: If the set of nodes and the dynamic parameter are invariant, then the result set of selecting is also invariant, which makes the fairness of selection irrelevant to the reliability of the selector. Once the dynamic parameter is given, the result of running the RS algorithm by anyone is the same. This property makes the RS algorithm impartial enough.

All properties are well supported by strict proofs. We also implement the proposed algorithm on personal computers, and experimental simulations show that our algorithm is efficient.

The rest of this paper is organized as follows. In Section 2, we briefly review some related works. Some preliminaries used in our proposed protocol are introduced in Section 3, and the system model and design goals of our scheme are formalized in

Section 4. We describe the details of our RS algorithm and DRBFT in Section 5 and give the proofs of properties in Section 6. The implementation results are shown in Section 7. Finally, some concluding remarks are given in Section 8.

2. Related work

In this section, we introduce some existing works which improve the efficiency of implementing PBFT.

PBFT [15] proposed by Castro et al. promises the practical application of the Byzantine fault tolerance algorithm. With the rise of large scale blockchain systems, however, the efficiency of PBFT hinders its further application. Therefore, researchers focus on improving the performance and efficiency of PBFT. The existing related schemes can be roughly divided into two categories, optimizing the algorithm structure [18–23] and reducing the number of nodes that execute the algorithm [16,17,24–26].

For the first goal, Yin et al. proposed a Byzantine fault tolerance protocol in 2003 [18]. The scheme reduces the replication costs and improves fault tolerance by separating the order request agreement from processing requests. In 2004, Kotla et al. presented HTBFT algorithm [19] with a general parallelizer module to enhance the throughput and efficiency of PBFT. In addition, Kotla designed an algorithm called Zyzzyva [20], which uses speculation mechanism to simplify BFT state machine replication while greatly reducing the replication overhead. Aiming at the situation that the efficiency of PBFT is easily affected by the primary's performance, Amir et al. proposed a novel Byzantine fault tolerant algorithm Prime [21]. This algorithm maintains the efficiency of the entire system when the primary is attacked and the performance decreases. The RBFT algorithm [22] proposed by Clement et al. improves the ability of PBFT to resist Byzantine malicious nodes and the consensus efficiency. Wood et al. proposed the ZZ consensus protocol [23] based on PBFT. In order to improve throughput and response time, ZZ reduces the number of execution replicas under normal conditions from $2f + 1$ to $f + 1$.

In another situation, DBFT [16] proposed by NEO uses DPoS to reduce the amount of nodes participating in PBFT consensus. Concretely, nodes in DBFT are divided into consensus nodes and normal nodes. Consensus nodes are in charge of communicating with each other to generate new blocks. Normal nodes are responsible for verifying new blocks. The key of DBFT is that all nodes in the blockchain vote for a certain number of nodes as consensus nodes. However, this step alone is still not enough. During the electing, each node votes for the node which can represent its own benefits best. Since the willingness will not change easily in a short term and the votes are cast with little enthusiasm, the nodes selected in each election are generally the same and not representative. Therefore, it is hard to guarantee the fairness and activity of systems just by using the voting idea of DPoS to select nodes.

To implement PBFT efficiently, Thunderchain [17] selects delegated nodes with Delegated Proof-of-Ability(DPoA) algorithm whose idea is based on DPoS. The selected process of DPoA is also relied on voting. The difference between DPoA and DPoS is that, in DPoA, the nodes whose performances of storage, network, bandwidth, latency etc. are better than others, are assigned more voting weight. Under such circumstances, elected nodes have better performances to ensure the stable operation of blockchain systems. However, there are some hidden threats by using this selecting way. Similar to PoW, the nodes with excellent performance have a high probability of being elected as representatives. This leads to a gradual convergence of high-performance equipment. Furthermore, the system loses fairness when the rights of accounting are held by several large consortium.

Similar to the idea of using randomly selecting representatives to improve PBFT, Gilad et al. presented a consensus protocol, Algorand [24,25], which utilizes verifiable random functions and a Byzantine agreement protocol to reach consensus on new blocks. To ensure the fairness of the system, the proposed functions select a set of verifier nodes with the randomness of hash functions. Specifically, all nodes use their secret keys to sign an information associated with the current block, and compute the hash value of the signature. The selected verifier nodes of the current round are the nodes whose hash value is less than a given parameter. In this way, their scheme guarantees the randomness and verifiability of the selection. However, verifying such signatures cost lots of computational resources. In particular, this method is hard to play well in asynchronous networks due to the network delay.

As a result, these existing works cannot ensure the fairness of selection while improving the efficiency of PBFT. As mentioned above, the activities of a distributed system depend on the participation of each node. Hence, the goal of our scheme is meaningful for the improvement of consensus algorithm in practice.

3. Preliminaries

In this section, we review some background knowledge about PBFT and probability distributions as well as hash functions.

3.1. PBFT

PBFT [15] is a state machine replication algorithm and first solves the fault tolerance issues of a distributed system in polynomial time. This protocol consists of three parties, a primary, $n - 1$ backups and several clients, and this system reaches consensus successfully when the number of invalid backups is less than f, where $n = 3f + 1$. The primary distributes a sequence number to the client-initiated request, and then reaches consensus with backups on whether this request will

be executed or not and the order of execution. There are five stages involved in this procedure, including *request*, *pre-prepare*, *prepare*, *commit* and *reply*.

In the *request* phase, the client broadcasts a REQUEST message and its signature $(m, sig(m)) = (REQUEST, o, t, c, sig(m))$ to the primary and backups, where o is a state machine operation, t is the current timestamp, and c is the identifier of the client.

In the *pre-prepare* phase, the primary rejects the REQUEST message if its signature is illegal. Otherwise, the primary assigns a sequence number d to the REQUEST message. Then, the primary broadcasts a PRE-PREPARE message $(PRE\text{-}PREPARE, v, d, H(m))$ and a corresponding signature sig_{PP} to backups, where v is the number of current view, and $H(m)$ is the hashed value of m.

In the *prepare* phase, each backup records the received information locally if they determine to accept the PRE-PREPARE message. Then, each backup transmits a PREPARE message $(PREPARE, v, d, H(m))$ and a signature sig_P of the PREPARE message to the primary as well as other backups.

In the *commit* phase, the primary and each backup verify the PREPARE message. After that, each backup records the received information locally and sends a COMMIT message $(COMMIT, v, d)$ with the corresponding signature sig_C to the primary and other backups if at least $2f$ received PREPARE message is validated.

In the *reply* phase, each backup rejects the request if the number of validated commit messages is less than $2f$. Otherwise, each backup executes the operation launched by the client, and transmits a REPLY message $(REPLY, v, t, c, i, r)$ and its signature sig_R to the client, where i is the number of the backup, and r is the operation result.

Finally, it means that the request launched by the client has reached consensus across the network if the client receives $f + 1$ identical REPLY messages. Otherwise, the client needs to retransmit the request.

As shown in Fig. 1, the primary, backup 1 and backup 2 can still reach a consensus on the client's request when backup 3 is a faulty node. However, the communication cost of PBFT is a huge burden when the number of nodes is large.

We only introduce a brief flow of the algorithm here. More details can be found in the original paper [15].

3.2. Probability distribution

It is worth to introduce two kinds of notions about probability distributions, statistical [27] and computational [28] indistinguishabilities. For two probability distributions, the first indistinguishability refers that no distinguishers could tell the difference between them. The second one means that no polynomial algorithm could distinguish them. The formal definitions are given below [29].

Definition 1 (*Statistical Indistinguishability*). Given two probability distributions **X** and **Y** on the set **D**, they are statistically indistinguishable if their statistical distance $\Delta(\mathbf{X}, \mathbf{Y})$ is negligible.

$$\Delta(\mathbf{X}, \mathbf{Y}) = \frac{1}{2}\sum_{d \in \mathbf{D}} |\Pr(\mathbf{X} = d) - \Pr(\mathbf{Y} = d)| \leqslant \varepsilon(n),$$

where n is the bit length of the elements in the set **D**.

Definition 2 (*Computational Indistinguishability*). Given two probability distributions **X** and **Y** on the set **D**, they are computationally indistinguishable if for any polynomial size circuit **A**, it has

$$Adv(\mathbf{A}) = |\Pr_{x \in \mathbf{X}}(\mathbf{A}(x) = 1) - \Pr_{x \in \mathbf{Y}}(\mathbf{A}(x) = 1)| \leqslant \varepsilon(n).$$

The symbol $\varepsilon(n)$ denotes a negligible function about a natural number n [30]. The definition of negligible functions is given below. This notion is important for complexity-based modern cryptographic algorithm design, especially for provably secure cryptographic schemes.

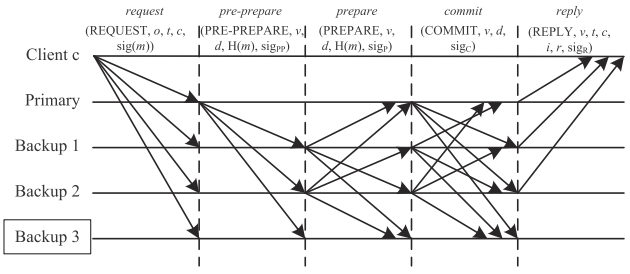

Fig. 1. PBFT flow diagram.

Definition 3 (*Negligible Function*). A function $\varepsilon(n)$ about a natural number n is negligible, if for any non-zero polynomial $poly(n)$, there exists a natural number N_0 such that

$$|\varepsilon(n)| < \left|\frac{1}{poly(n)}\right|$$

holds for all $n > N_0$.

Based on the above definition, we show several lemmas about negligible functions, and their proofs can be found in [30].

Lemma 1. *For an arbitrary positive polynomial $p(n)$, the function $\varepsilon(n) = p(n)/2^n$ is negligible.*

Lemma 2. *If $\varepsilon(n)$ is a negligible function and $p(n)$ is an arbitrary positive polynomial, then the function $\varepsilon^*(n) = \varepsilon(n)p(n)$ is negligible.*

Lemma 3. *If both $\varepsilon_1(n)$ and $\varepsilon_2(n)$ are negligible functions, then the function $\varepsilon(n) = \varepsilon_1(n) + \varepsilon_2(n)$ is negligible.*

3.3. Hash function

A hash function is a deterministic function that maps an arbitrary bit string into a hashed value with fixed bit length [31]. In the following, we first introduce four desirable properties of hash functions [31]. Let the symbol H denote a hash function, and the symbol $|H|$ denote the fixed output bit length of this hash function.

(1) *Mixing-transformation*: The hashed value $H(x)$ of an arbitrary input x should be computationally indistinguishable from a uniform binary string in the interval $[0, 2^{|H|} - 1]$.
(2) *Collision resistance*: Finding two different inputs a and b that satisfy $H(a) = H(b)$ should be computationally infeasible.
(3) *Pre-image resistance*: Finding a bit string x that satisfies $H(x) = h$ for a given hashed value h should be computationally infeasible.
(4) *Practical efficiency*: The hashed value of a given input string x should be calculated in a small-degree polynomial time in the size of x.

In fact, the hash function is used as a pseudo-random function to generate random number in our RS algorithm. In order to facilitate the performance analysis of the scheme, the following assumption made based on the first property of hash functions.

Assumption 1. Suppose the output of an ideal hash H is $x_1x_2 \ldots x_n$, then there are

(1) Each bit of the output of the hash function H is independent, i.e.,

$$\Pr(x_i|x_j) = \Pr(x_i), 1 \leqslant i, j \leqslant n \text{and} i \neq j.$$

(2) The value of each bit of the hash function H is random, i.e.,

$$\Pr(x_i = 0) = \Pr(x_i = 1) = 1/2, 1 \leqslant i \leqslant n.$$

4. System model and design goals

In this section, we formalize the system model and identify the design goals considered of our scheme.

4.1. System model

As shown in Fig. 2, there are three parties in our system model, including candidates, councillors, and normal nodes.

- *Candidates*: The candidates are elected from the whole nodes in a blockchain system at the beginning of the consensus process. Since they could represent the interests of most nodes, they will form an alternative pool of nodes that perform PBFT.
- *Councillors*: In each execution cycle, the councillors are selected from the candidates with the RS algorithm. They are responsible for implementing PBFT algorithm to achieve consensus. The primary and backups in PBFT are all selected from the councillors.
- *Normal nodes*: All nodes except candidates and councillors are called normal nodes. They are responsible for recording the results of consensus and verifying the information as assistants.

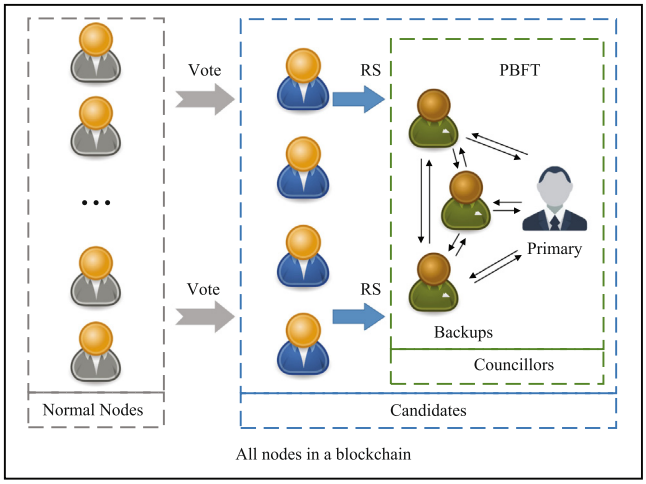

Fig. 2. The system model under consideration.

4.2. Design goals

Under the system model, the design goals of our proposed scheme are considered as follows.

- *Fairness and impartiality*. In order to improve the efficiency of achieving consensus in blockchains, it is practical to select representative nodes. However, ensuring the fairness and impartiality of the selection process is also critical to maintain the activity of systems. On the one hand, the selected nodes perform better than other nodes in executing the consensus algorithm. On the other hand, if performance is the only consideration, then the selected nodes will always be the best performing class of nodes, which will reduce the activity of nodes.
- *Byzantine faults tolerance*. Under the asynchronous network environment, the proposed consensus protocol should have the ability to tolerate the Byzantine faults. In this way, this protocol is more suitable for wider application scenarios.
- *Efficiency*. The proposed protocol including the RS algorithm should be as efficient as possible, which is our original intention to improve PBFT.

5. Our construction

In this section, we first introduce the RS algorithm which is a significant part of DRBFT, and then present the details of DRBFT protocol.

5.1. RS algorithm

In this subsection, we first introduce a random selection algorithm based on the hash function, the purpose of which is to select elements from a given set of elements randomly. It is worth to mention that, the hash function used here has excellent properties that we describe in Section 2. We give a detailed description of the algorithm as follows, and the corresponding pseudo-code description is shown in Algorithm 1.

Algorithm 1. RS Algorithm

Input: A dynamic parameter $seed$, a set $\mathbf{P}$ with P different elements, i.e., $\mathbf{P} = \{a_1, a_2, ...a_P\}$ and the target number of selection m, where $0 < m \leq P$.

Initialize an empty set as the set of selected elements $\mathbf{p}$.
for $i = 1, \cdots, m$ **do**
  Compute $seed = H(seed)$.
  Compute $s = seed(\mathrm{mod}(P - i + 1)) + 1$.
  Set $\mathbf{p} = \mathbf{p} \bigcup \{a_s\}$.
  Set $\mathbf{P} = \mathbf{P} - \{a_s\}$.
  Renumber the set $\mathbf{P}$, $\mathbf{P} = \{a_1, a_2, ...a_{P-i}\}$.
end

Output: The set $\mathbf{p}$.

The RS algorithm could be described as randomly selecting m elements $\mathbf{p} = \{p_1, p_2, \ldots, p_m\}$ from a set $\mathbf{P} = \{a_1, a_2, \ldots a_P\}$ with P elements, while ensuring that the selected elements are different from each other. It is worth to mention that the selected element a_i is reformulated into p_j in order to describe the algorithm clearly, i.e., $i = 1, \ldots, m, p_i \in \mathbf{P}$. The set $\mathbf{A}_P$ is defined as follows,

$$\mathbf{A}_P = \{a_P = \{p_1, \ldots, p_m\} | \forall i = 1, \ldots, m, p_i \in \mathbf{P} \text{ and } p_1, \ldots, p_m \text{ are all different}\}.$$

As a random seed, the initial value of the dynamic parameter *seed* is given before selecting members. To ensure the fairness and unpredictability of selection, the initial value of *seed* should be a string associated to the current or previous round's information, such as $seed = H(h)$, where h is the block header of the previous block.

In practice, it is hard to select the set $\mathbf{p} = \{p_1, \ldots, p_m\}$ at once. Therefore, we accomplish this procedure in an iterative way. Specifically, we first select p_1 from the set $\mathbf{P} = \{a_1, \ldots, a_P\}$ and put it into the set $\mathbf{p}$, where p_1 is the s-th member in the set $\mathbf{P}$ and $s = H(seed)(\mathrm{mod}(P)) + 1$. Then, we update the dynamic parameter as $seed = H(seed)$ and calculate $s = H(seed)(\mathrm{mod}(P-1)) + 1$. After that, we select the s-th member as p_2 from the set $\mathbf{P} - \{p_1\} = \{a_1, \ldots, a_P\} - \{p_1\}$ and put it into the set $\mathbf{p}$, and so on. Finally, we select the s-th member as p_m from the set $\mathbf{P} - \{p_1, \ldots, p_{m-1}\} = \{a_1, \ldots, a_P\} - \{p_1, \ldots, p_{m-1}\}$ and put it into the set $\mathbf{p}$ after computing $seed = H(seed)$ as well as $s = H(seed)(\mathrm{mod}(P-m+1)) + 1$. As a result, we get a set of selected members $\mathbf{p} = \{p_1, \ldots, p_m\}$.

Correctness: Notice that, for any $H(seed) \in \mathbb{Z}$, it has $0 < H(seed)(\mod(P)) \leqslant P - 1$. Furthermore, it can derive that $0 < H(seed)(\mathrm{mod}(P)) + 1 \leqslant P$. Hence, the s-th member of the set $\mathbf{P}$ is meaningful. In order to ensure that the selected members are different, the selected member is removed from the set $\mathbf{P}$ after each selection.

It is hoped that the final output set $\mathbf{p}$ will be subject to a uniform distribution on the set $\mathbf{A}_P$, i.e., $\forall a_P = \{p_1, \ldots, p_m\} \in \mathbf{A}_P$, there is

$$\Pr[\mathbf{U}(P, m) = a_P] = \frac{1}{|\mathbf{A}_P|} = \frac{1}{\binom{P}{m}} = \frac{1}{\frac{P!}{m!(P-m)!}} = \frac{m!(P-m)!}{P!}.$$

However, this condition is hard to be satisfied. In the practical schemes of cryptography, statistical indistinguishability belongs to strong cryptographic primitives, which is enough to ensure the security of schemes. Let $W(P, m)$ denote the probability distribution of the output of our RS algorithm. In the next section, we will prove that the probability distribution $W(P, m)$ is statistically indistinguishable from the uniform distribution $\mathbf{U}(P, m)$. This conclusion ensures that the selection process is sufficiently random.

5.2. DRBFT scheme

In this part, we elaborate on the DRBFT in four steps.

Algorithm 2. Voting Algorithm

Input: All nodes in the blockchain $\{a_1, \cdots, a_n\}$.
Initialize $Vote_i = 0, i = 1, \cdots, n$.
Initialize an empty set **Vote**.
for $i = 1, \cdots, n$ **do**
  Compute $Vote_i = \sum_{j=1}^{n} w_j Vote_{ji}$.
end
Sort the vote number of each node from high to low to get the set **Vote**.
Output: The set of votes obtained by all nodes **Vote**.

Algorithm 3. Getting M Candidates

Input: All nodes in the blockchain $\{a_1, \cdots, a_n\}$, the set of votes obtained by all nodes **Vote** and the constant number of candidates M.

Initialize two empty sets **Cand** and **Nor**.
for $i = 1, \cdots, n$ **do**
  if *the position of* $Vote_i$ *in the set* **Vote** *is the top* M, **then**
    Set $\mathbf{Cand} = \mathbf{Cand} \bigcup \{a_i\}$.
  end
  else
    Set $\mathbf{Nor} = \mathbf{Nor} \bigcup \{a_i\}$.
  end
end

Output: The set of candidates **Cand** and the set of normal nodes **Nor**.

Step1: Number all participating nodes to ensure that each node is identifiable.

Step2: Run a voting algorithm with all nodes as inputs to get a list of M candidates. Voting algorithms can fairly select a certain number of nodes according to the wishes of most nodes, such as the voting mechanism in DPoS [13]. As shown in Algorithm 2, we introduce an intuitive voting algorithm. The number of votes received by each node is denoted as $Vote_i$. If a_j votes for a_i, set $Vote_{ji} = 1$ or $Vote_{ji} = 0$ otherwise. In addition, the parameter w_i represents the voting weight of each node. In order to ensure the consensus efficiency, we set the number of candidates as a constant. Therefore, as shown in Algorithm 3, the selected candidate is the node with the top M votes received.

Step3: Run the RS algorithm with M candidates as inputs to select N councillors. As described in the last subsection, the RS algorithm selects a given number of members from a set of members in a random way. Take M candidates, parameter N and a string *seed* as the inputs of the RS algorithm and let the results of selection be the councillors. The dynamic parameter *seed* is the hash value of the content of the last block. In this way, *seed* is not only random, verifiable, but also uniform in the whole blockchain nodes.

Step4: N councillors execute PBFT to create consensus and generate new blocks. Initially, PBFT selects a primary from councillors, and the rest councillors become backups. Then, the primary and backups exchange information through five stages to achieve consensus on the requests of non-councillors nodes. The primary and backups locally store the requests that reach consensus as new generated blocks. Finally, the new block is broadcasted to all non-councillors nodes for synchronization. If the consensus fails due to the councillors, the RS algorithm will be re-executed to select new councillors and the consensus process will be restarted.

6. Security proof

In this section, we analyze the properties of our RS algorithm, including unpredictability, uniform distribution and impartiality. Firstly, we transform the practical problems into a mathematical model. Selecting m members from the set $\mathbf{P} = \{a_1, a_2, \ldots a_P\}$ with P numbered nodes is equal to the situation that selecting m members from the set $\mathbf{P} = \{1, 2, \ldots, P\}$ with P positive integers. In order to prove more intuitively, we introduce two intermediate probability distributions $\mathbf{X}$ and $\mathbf{V}(P, m)$ on the set $\mathbf{P}$, respectively.

For any $j \in \mathbf{T} = \{0, 1, 2, \ldots, 2^n - 1\}$, let $\mathbf{X}(j) = j(\mathrm{mod}P) + 1$, where P refers to the number of members in the set $\mathbf{P} = \{1, 2, \ldots, P\}$. Notice that, for any $j \in \mathbf{P}$, it has $1 \leqslant j(\mathrm{mod}P) + 1 \leqslant P$ which could be inferred from $0 \leqslant j(\mathrm{mod}P) \leqslant P - 1$ easily. Hence, $\mathbf{X}$ is indeed a distribution on the set $\mathbf{P}$. Furthermore, we have the following lemma.

Lemma 4. *The distribution* $\mathbf{X}$ *on the set* $\mathbf{P}$ *is statistically indistinguishable from the uniform distribution* $\mathbf{U}$ *on the set* $\mathbf{P}$.

Proof. Without loss of generality, let $2^n = kP + r$ where $0 \leqslant r \leqslant P - 1$ is the non-negative least residue of the division of 2^n by P, and $k = [2^n/P] \leq 2^n/P$ is the corresponding incomplete quotient. For any $i \in \mathbf{P}$, there is

$$\begin{aligned} \Pr[\mathbf{X} = i \in \mathbf{P}] &= \Pr[\mathbf{X}(j) = j(\mathrm{mod}\ P) + 1 = i : j \in \mathbf{T}] \\ &= \Pr[j(\mathrm{mod}\ P) + 1 = i : j \in \mathbf{T}] \\ &= \Pr[j(\mathrm{mod}\ P) = i - 1 : j \in \mathbf{T}]. \end{aligned}$$

As described, for any $i \in \mathbf{P}$, there is $0 \leqslant j(\mathrm{mod}\ P) = i - 1 \leqslant P - 1$.

If $0 \leqslant j(\mathrm{mod}P) = i-1 \leqslant r-1$, i.e., $1 \leqslant j(\mathrm{mod}\ P)+1 = i \leqslant r$, the number of j in $\mathbf{T}$ such that $j(\mathrm{mod}\ P) = i-1$ is $k+1 = [2^n/P]+1 > 2^n/P$. Hence, there is

$$\Pr(\mathbf{X} = i \in \mathbf{P}) = \Pr[j(\mathrm{mod}\ P) = i-1 : j \in \mathbf{T}] = \frac{k+1}{2^n} > \frac{1}{P}. \tag{1}$$

If $r \leqslant j(\mathrm{mod}\ P) = i-1 \leqslant P-1$, i.e., $r+1 \leqslant j(\mathrm{mod}\ P) = i \leqslant P$, the number of j in $\mathbf{T}$ such that $j(\mathrm{mod}\ P) = i-1$ is $k = [2^n/P] \leq 2^n/P$. So, it has

$$\Pr(\mathbf{X} = i \in \mathbf{P}) = \Pr[j(\mathrm{mod}\ P) = i-1 : j \in \mathbf{T}] = \frac{k}{2^n} \leqslant \frac{1}{P}. \tag{2}$$

According to the Eqs. (1) and (2), the statistical distance between the two distributions $\mathbf{X}$ and $\mathbf{U}$ are calculated as follows.

$$\begin{aligned}
&\Delta(\mathbf{X},\mathbf{U}) = \tfrac{1}{2}\sum_{i\in\mathbf{P}}|\Pr(\mathbf{X}=i) - \Pr(\mathbf{U}=i)| \\
&= \tfrac{1}{2}\sum_{1\leqslant i\leqslant r}|\Pr(\mathbf{X}=i) - \Pr(\mathbf{U}=i)| + \tfrac{1}{2}\sum_{r+1\leqslant i\leqslant P}|\Pr(\mathbf{X}=i) - \Pr(\mathbf{U}=i)| \\
&= \tfrac{1}{2}\sum_{1\leqslant i\leqslant r}|\tfrac{k+1}{2^n} - \tfrac{1}{P}| + \tfrac{1}{2}\sum_{r+1\leqslant i\leqslant P}|\tfrac{k}{2^n} - \tfrac{1}{P}| \\
&= \tfrac{1}{2}[r(\tfrac{k+1}{2^n} - \tfrac{1}{P}) + (P-r)(\tfrac{1}{P} - \tfrac{k}{2^n})] \\
&= \tfrac{1}{2}[r\tfrac{kP+P-2^n}{2^nP} + (P-r)\tfrac{2^n-kP}{2^nP}] \\
&= \tfrac{1}{2}[r\tfrac{P-r}{2^nP} + (P-r)\tfrac{r}{2^nP}] \\
&= \tfrac{r(P-r)}{2^nP}.
\end{aligned}$$

Since $r(P-r) \leqslant [r+(P-r)]^2/4 = P^2/4$, we can deduce that $\Delta(\mathbf{X},\mathbf{U}) \leqslant P/2^{n+2}$. As a result, the distance between the two distributions $\mathbf{X}$ and $\mathbf{U}$ is negligible according to Lemma 1. □

In addition, we give the description of another intermediate probability distribution $\mathbf{V}(P,m)$ on the set $\mathbf{P}$. First, randomly select a member p_1 from the set $\mathbf{P} = \{1,2,\ldots,P\}$ (the probability that each member will be selected is $1/P$). Then, randomly select a member p_2 from the set $\mathbf{P} - \{p_1\} = \{1,2,\ldots,P\} - \{p_1\}$ (the probability that each member will be selected is $1/(P-1)$) and so on. The output set $\mathbf{p} = \{p_1, p_2 \ldots, p_m\}$ will be obtained until p_m has been selected from the set $\mathbf{P} - \{p_1, p_2, \ldots, p_{m-1}\} = \{1,2,\ldots,P\} - \{p_1, p_2, \ldots, p_{m-1}\}$. We prove that this way of selecting is equal to the way of randomly selecting at once.

Lemma 5. *The probability distribution $\mathbf{V}(P,m)$ is equal to the uniform distribution $\mathbf{U}$.*

Proof. $\mathbf{V}(P,m)$ is also a distribution on the set $\mathbf{A}_P$. Hence, we just need to prove that $\forall a_P = \{p_1, p_2, \ldots, p_m\} \in \mathbf{A}_P$, there is

$$\Pr[\mathbf{U}(P,m) = a_P] = \Pr[\mathbf{V}(P,m) = a_P] = \frac{m!(P-m)!}{P!}.$$

Notice that

$$\begin{aligned}
&\Pr[\mathbf{V}(P,m) = a_P = \{p_1, p_2, \ldots, p_m\}] \\
&= m!\Pr[\text{select } p_1 \text{ for the first time} \wedge \ldots \wedge \text{ select } p_m \text{ for the } m\text{-th time}] \\
&= m!\frac{1}{\text{Number of all options that select } p_1 \ldots p_m \text{ from } \mathbf{P}} \\
&= m!\frac{1}{P\times\ldots\times(P-m+1)} \\
&= m!\frac{1\times2\times\ldots\times(P-m)}{1\times2\times\ldots\times(P-m)\times P\times\ldots\times(P-m+1)} \\
&= \frac{m!(p-m)!}{P!} \\
&= \Pr[\mathbf{U}(P,m) = a_P].
\end{aligned}$$

Remark: There is "$m!$" on the right side of the first equation since $\Pr[\mathbf{V}(P,m) = a_P = \{p_1, p_2, \ldots, p_m\}]$ does not consider the order, but $\Pr[$. select p_1 for the first time $\wedge \ldots \wedge$ select p_m for the m-th time $]$ does. The symbol $m!$ denotes the factorial of m. □.

Next, we give a useful lemma about the statistical indistinguishability for joint probability distributions.

Lemma 6. *Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be two nonempty sets, if there exists a polynomial $p(n)$ such that $|\mathbf{D}_1| \leqslant p(n)$ and $|\mathbf{D}_2| \leqslant p(n)$, then the probability distributions $\mathbf{X}_1$ and $\mathbf{X}_2$ on the set $\mathbf{D}_1$ are statistically indistinguishable, the probability distributions $\mathbf{Y}_1$ and $\mathbf{Y}_2$ on the set $\mathbf{D}_2$ are statistically indistinguishable, $\mathbf{X}_1$ is statistically independent from $\mathbf{Y}_1, \mathbf{X}_2$ is statistically independent from $\mathbf{Y}_2$, then the joint probability distribution $(\mathbf{X}_1, \mathbf{Y}_1)$ on the set $\mathbf{D}_1 \times \mathbf{D}_2$ is statistically indistinguishable from $(\mathbf{X}_2, \mathbf{Y}_2)$.*

Proof. According to Definition 1, there are two negligible functions $\varepsilon_1(n)$ and $\varepsilon_2(n)$ such that

$$\Delta(\mathbf{X}_1,\mathbf{X}_2)=\frac{1}{2}\sum_{i\in\mathbf{D}_1}|\Pr(\mathbf{X}_1=i)-\Pr(\mathbf{X}_2=i)|\leqslant\varepsilon_1(n),$$

$$\Delta(\mathbf{Y}_1,\mathbf{Y}_2)=\frac{1}{2}\sum_{j\in\mathbf{D}_2}|\Pr(\mathbf{Y}_1=j)-\Pr(\mathbf{Y}_2=j)|\leqslant\varepsilon_2(n).$$

Hence, $\forall i\in\mathbf{D}_1$, it has

$$|\Pr(\mathbf{X}_1=i)-\Pr(\mathbf{X}_2=i)|\leqslant\sum_{i\in\mathbf{D}_1}|\Pr(\mathbf{X}_1=i)-\Pr(\mathbf{X}_2=i)|=2\Delta(\mathbf{X}_1,\mathbf{X}_2)\leqslant2\varepsilon_1(n).$$

It means that there exists a negligible function $\varepsilon^*_{1i}(n)$ such that

$$\Pr(\mathbf{X}_1=i)=\Pr(\mathbf{X}_2=i)+\varepsilon^*_{1i}(n).$$

Similarly, there exists a negligible function $\varepsilon^*_{2j}(n)$ such that

$$\Pr(\mathbf{Y}_1=j)=\Pr(\mathbf{Y}_2=j)+\varepsilon^*_{2j}(n).$$

Then, we calculate the statistical distance between $(\mathbf{X}_1,\mathbf{Y}_1)$ and $(\mathbf{X}_2,\mathbf{Y}_2)$ on the set $\mathbf{D}_1\times\mathbf{D}_2$, where $(i,j)\in\mathbf{D}_1\times\mathbf{D}_2$.

$$\begin{aligned}
&\Delta[(\mathbf{X}_1,\mathbf{Y}_1),(\mathbf{X}_2,\mathbf{Y}_2)]\\
=&\tfrac{1}{2}\textstyle\sum|\Pr(\mathbf{X}_1=i\wedge\mathbf{Y}_1=j)-\Pr(\mathbf{X}_2=i\wedge\mathbf{Y}_2=j)|\\
=&\tfrac{1}{2}\textstyle\sum|\left[\Pr(\mathbf{X}_2=i)+\varepsilon^*_{1i}(n)\right]\left[\Pr(\mathbf{Y}_2=j)+\varepsilon^*_{2j}(n)\right]-\Pr(\mathbf{X}_2=i)\Pr(\mathbf{Y}_2=j)|\\
=&\tfrac{1}{2}\textstyle\sum|\varepsilon^*_{1i}(n)\Pr(\mathbf{Y}_2=j)+\varepsilon^*_{2j}(n)\Pr(\mathbf{X}_2=i)+\varepsilon^*_{1i}(n)\varepsilon^*_{2j}(n)|\\
\leqslant&\tfrac{1}{2}\textstyle\sum\left[|\varepsilon^*_{1i}(n)|\Pr(\mathbf{Y}_2=j)+|\varepsilon^*_{2j}(n)|\Pr(\mathbf{X}_2=i)+|\varepsilon^*_{1i}(n)\varepsilon^*_{2j}(n)|\right]\\
\leqslant&\tfrac{1}{2}\textstyle\sum\left[|\varepsilon^*_{1i}(n)|+|\varepsilon^*_{2j}(n)|+|\varepsilon^*_{1i}(n)\varepsilon^*_{2j}(n)|\right]\\
\leqslant&\tfrac{1}{2}\textstyle\sum\{\max_{i\in\mathbf{D}_1}\left[|\varepsilon^*_{1i}(n)|\right]+\max_{j\in\mathbf{D}_2}\left[|\varepsilon^*_{2j}(n)|\right]+\max_{i\in\mathbf{D}_1,j\in\mathbf{D}_2}\left[|\varepsilon^*_{1i}(n)\varepsilon^*_{2j}(n)|\right]\}\\
\leqslant&\tfrac{1}{2}p^2(n)\{\max_{i\in\mathbf{D}_1}\left[|\varepsilon^*_{1i}(n)|\right]+\max_{j\in\mathbf{D}_2}\left[|\varepsilon^*_{2j}(n)|\right]+\max_{i\in\mathbf{D}_1,j\in\mathbf{D}_2}\left[|\varepsilon^*_{1i}(n)\varepsilon^*_{2j}(n)|\right]\}\\
=&\tfrac{1}{2}p^2(n)\{\max_{i\in\mathbf{D}_1}\left[|\varepsilon^*_{1i}(n)|\right]+\max_{j\in\mathbf{D}_2}\left[|\varepsilon^*_{2j}(n)|\right]+\max_{i\in\mathbf{D}_1}\left[|\varepsilon^*_{1i}(n)|\right]\max_{j\in\mathbf{D}_2}\left[|\varepsilon^*_{2j}(n)|\right]\}\\
\leqslant&\tfrac{1}{2}p^2(n)\{2\max_{i\in\mathbf{D}_1}\left[|\varepsilon^*_{1i}(n)|\right]+\max_{j\in\mathbf{D}_2}\left[|\varepsilon^*_{2j}(n)|\right]\}.
\end{aligned}$$

It is clear that $1/2p^2(n)\left[2|\varepsilon^*_{1i}(n)|+|\varepsilon^*_{2j}(n)|\right]$ is a negligible function according to Lemmas 2 and 3. □

Based on the above lemmas, we give the following theorems to support the properties of the RS algorithm.

Theorem 1. *The distributions* $\mathbf{W}(P,m)$ *and* $\mathbf{V}(P,m)$ *on the set* $\mathbf{A}_P$ *are statistically indistinguishable.*

Proof. We use mathematical induction to prove this theorem.

When $m=1$, in fact, the distribution $\mathbf{X}$ in Lemma 4 is equal to the distribution $\mathbf{W}(P,1)$ according to Assumption 1, and the distribution $\mathbf{U}$ is equal to the distribution $\mathbf{V}(P,1)$. Hence, $\mathbf{W}(P,1)$ is statistically indistinguishable from $\mathbf{V}(P,1)$ on the basis of Lemma 4.

Suppose $\mathbf{W}(P,m-1)$ is statistically indistinguishable from $\mathbf{V}(P,m-1)$ according to mathematical induction, i.e.,

$$\mathbf{W}(P,m)=(\mathbf{W}(P,m-1),\mathbf{W}(P-m+1,1)),$$

$$\mathbf{V}(P,m)=(\mathbf{V}(P,m-1),\mathbf{V}(P-m+1,1)).$$

According to Lemma 4, the distribution $\mathbf{W}(P-m+1,m)$ is statistically indistinguishable from $\mathbf{V}(P-m+1,1)$. In addition, the probability experiments of $\mathbf{W}(P,m-1)$ and $\mathbf{V}(P,m-1)$ are independent. The probability experiments of $\mathbf{W}(P-m+1,1)$ and $\mathbf{V}(P-m+1,1)$ are also independent. Therefore, the distributions $\mathbf{W}(P,m-1)$ is statistically independent from $\mathbf{V}(P,m-1)$, so do $\mathbf{W}(P-m+1,1)$ and $\mathbf{V}(P-m+1,1)$. As a result, the distribution $\mathbf{W}(P,m)=(\mathbf{W}(P,m-1),\mathbf{W}(P-m+1,1))$ is statistically indistinguishable from $\mathbf{V}(P,m)=(\mathbf{V}(P,m-1),\mathbf{V}(P-m+1,1))$ on the basis of Lemma 6. □

Theorem 2. *The distributions* $\mathbf{W}(P,m)$ *and* $\mathbf{U}(P,m)$ *on the set* $\mathbf{A}_P$ *are statistically indistinguishable.*

Proof. The statistical distance between $\mathbf{W}(P,m)$ and $\mathbf{U}(P,m)$ is calculated as follows.

$$\begin{aligned}
&\Delta[\mathbf{W}(P,m),\mathbf{U}(p,m)]\\
&=\tfrac{1}{2}\sum|\Pr(\mathbf{W}(P,m)=a_P)-\Pr(\mathbf{U}(P,m)=a_P)|\\
&=\tfrac{1}{2}\sum|\Pr(\mathbf{W}(P,m)=a_P)-\Pr(\mathbf{V}(P,m)=a_P)+\Pr(\mathbf{V}(P,m)=a_P)-\\
&\Pr(\mathbf{U}(P,m)=a_P)|\\
&\leqslant\tfrac{1}{2}[\sum|\Pr(\mathbf{W}(P,m)=a_P)-\Pr(\mathbf{V}(P,m)=a_P)|+\sum|\Pr(\mathbf{V}(P,m)=a_P)-\\
&\Pr(\mathbf{U}(P,m)=a_P)|]\\
&=\tfrac{1}{2}\{2\Delta[\mathbf{W}(P,m),\mathbf{V}(P,m)]+2\Delta[\mathbf{V}(P,m),\mathbf{U}(P,m)]\}\\
&=\Delta[\mathbf{W}(P,m),\mathbf{V}(P,m)]+\Delta[\mathbf{V}(P,m),\mathbf{U}(P,m)],
\end{aligned}$$

where $a_P\in\mathbf{A}_P$. The statistical distance $\Delta[\mathbf{V}(P,m),\mathbf{U}(P,m)]=0$ and $\Delta[\mathbf{W}(P,m),\mathbf{V}(P,m)]$ is negligible according to Lemma 5 and Theorem 1, respectively. Hence, The statistical distance $\Delta[\mathbf{W}(P,m),\mathbf{U}(p,m)]$ is negligible. □

Finally, we analyze the properties of RS algorithm based on the above conclusions.

(1) According to Theorem 2, it is clear that the distribution of the output of RS algorithm is statistically indistinguishable from the *uniform distribution*. Hence, the probability that each member being selected is equal.
(2) In particular, the selector has no permission to select parameters. This ensures that no matter who executes RS algorithm, the results are the same. Therefore, RS algorithm has the *impartiality*.
(3) Furthermore, the RS algorithm's outputs are *unpredictable* since the hash value of the dynamic parameter is unpredictable.

In the next section, we discuss the efficiency of RS algorithm under practical experiments.

7. Implementations

In this section, we evaluate the performance of RS algorithm from different aspects, including unpredictability, uniform distribution, impartiality and efficiency. In practice, SHA-256 [32] is adopted as the ideal hash function in the RS algorithm. The proposed algorithm is validated on a personal computer with go language. The environment parameters are listed as follows:

- CPU: Intel(R) Pentium(R) 2.60 GHz
- RAM: 2.00 GB
- OS: Windows 7

7.1. Unpredictability

For a fixed pair of parameters (P,m), the RS algorithm is executed one hundred times with dynamic parameter *seed* regenerated for each implementation. The difference of running results proves that the outputs of RS algorithm are hard to be predicted.

7.2. Uniform distribution

To verify that the distribution of the outputs of RS algorithm is statistically indistinguishable from the uniform distribution, we select five pairs of (P,m) and run the Algorithm 10000 times under each pair, where the dynamic parameter *seed* is updated each time. The results show that selected times of each element in the set $\mathbf{P}$ is approximately equal. For example, as Fig. 3 shows, each node is roughly selected 2500 times when $P=20$ and $m=5$. The corresponding theoretical value is $5/20\times 10000=2500$ if it is uniform distribution. Therefore, this property of RS algorithm is verified from a statistical perspective.

7.3. Impartiality

Similarly, in this part, the RS algorithm is repeatedly executed with the same inputs, i.e., P, m and *seed* remain unchanged. The result of multiple runs is exactly the same, which means that the result of selection depends on the initial parameters, not the selector. Therefore, everyone could monitor the procedure of selection easily, which enhances the transparency and impartiality of systems.

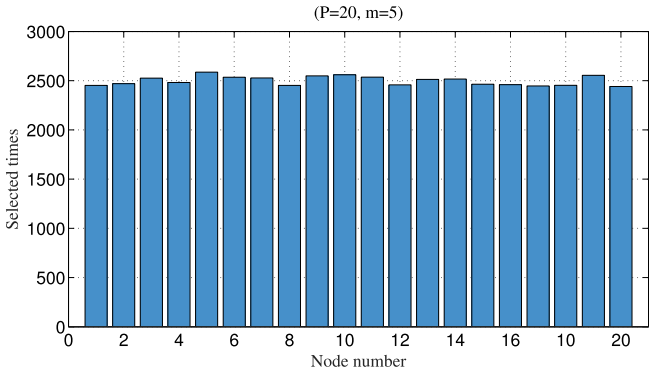

Fig. 3. The statistical results of RS algorithm.

7.4. Efficiency

In order to test the efficiency of the scheme, the RS algorithm is executed 100 times under different parameters. As shown in Fig. 4, we set $P = 100000, 200000, 500000, 1000000$ and $m = 50, 100, 150, 200, 250, 300$ respectively. The implementation results demonstrate that our scheme is highly practical. Even if there are 1000000 nodes in the system, it takes only 251.98 ms for the proposed scheme to select 300 representative members. In particular, with the growth of the number of nodes, the running time of RS algorithm increases almost linearly which is more pragmatic.

7.5. Comparison with related works

We compare DRBFT with traditional consensus protocols PoW, DPoS, PBFT, and PBFT-based consensus protocols DBFT and DPoA+PBFT. The comparison results are shown in Table 1. It must be pointed out that the efficiency is considered in the network composed of large-scale nodes. It is hard for PoW and DPoS to tolerate the Byzantine faults. PBFT is inefficient when the

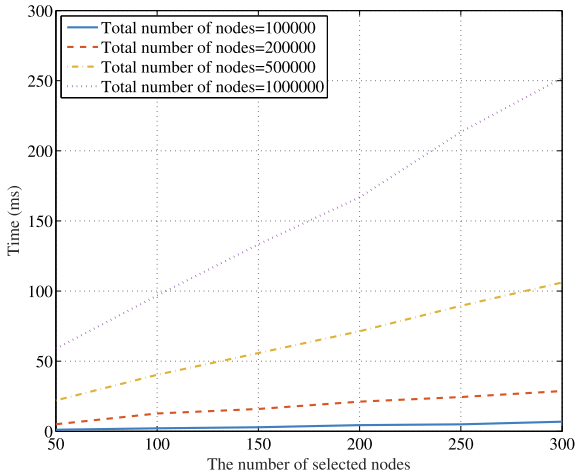

Fig. 4. The running time of RS algorithm.

Table 1
Comparison with related works.

Consensus protocol	Degree of de-centralization	System activity	Efficiency	Byzantine fault tolerance	Waste of resources
PoW	Total	Low	Medium	No	High
DPoS	High	Medium	High	No	Low
PBFT	Medium	High	Low	Yes	Low
DBFT	Medium	Medium	High	Yes	LoW
DPoA+PBFT	Medium	Low	High	Yes	High
DRBFT	Medium	High	High	Yes	Low

number of involved nodes is large. DBFT and DPoA+PBFT will decrease the system activity since using only voting mechanisms does not guarantee sufficient fairness. Besides, in Pow and DPoA+PBFT, there is a waste of resources because nodes spend extra resources to compete for accounting rights. As a result, considering comprehensively, our scheme is more practical than others.

8. Conclusion

In this paper, we present a random selection algorithm RS based on hash functions to select representative nodes from all nodes in the blockchain. The distribution of the outputs of RS algorithm is proven to be statistically indistinguishable from the uniform distribution, which ensures the fairness of selection. In addition, the initiative and activity of blockchain systems are guaranteed by the unpredictability and impartiality of RS algorithm. Furthermore, pair with the voting mechanism and PBFT, we propose a consensus protocol DRBFT to solve the issue of inefficiency in the blockchain with large-scale nodes under the asynchronous network environment. The implementations illustrate that our scheme is efficient and practical.

CRediT authorship contribution statement

Yu Zhan: Software, Validation, Formal analysis, Writing - original draft. **Baocang Wang:** Conceptualization, Methodology, Supervision. **Rongxing Lu:** Writing - review & editing. **Yong Yu:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors thank the anonymous reviewers for their valuable suggestions and comments, which help the authors improve their work. This work is supported by the National Natural Science Foundation of China under Grant Nos. U19B2021, 61972457, the National Cryptography Development Fund under Grant Nos. MMJJ20180111, MMJJ20180219, the Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04, the Key Technologies R&D Program of He'nan Province under Grant Nos. 212102210084, 192102210295, and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province.

References

[1] D. Brandon, The blockchain: The future of business information systems, International Journal of the Academic Business World 10 (2) (2016) 33–40.
[2] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system..
[3] C. Lee, Litecoin-open source p2p digital currency, https://litecoin.org/, accessed 2019, 2011..
[4] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, self-published paper, August 19..
[5] P. Vasin, Blackcoin's proof-of-stake protocol v2, https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf, accessed 2019, 2014..
[6] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, J. Kishigami, Blockchain contract: Securing a blockchain applied to smart contracts, in: 2016 IEEE International Conference on Consumer Electronics (ICCE), IEEE, 2016, pp. 467–468..
[7] V. Buterin, A next-generation smart contract and decentralized application platform, White Paper 3 (2014) 37.
[8] K. Biswas, V. Muthukkumarasamy, Securing smart cities using blockchain technology, in: 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2016, pp. 1392–1393..
[9] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec, Using blockchain for medical data access and permission management, in: 2nd International Conference on Open and Big Data (OBD), IEEE, 2016, pp. 25–30.
[10] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: a survey, International Journal of Web and Grid Services 14 (4) (2018) 352–375.
[11] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper 151 (2014) (2014) 1–32.
[12] C. Cachin, M. Vukolić, Blockchain consensus protocols in the wild, arXiv preprint arXiv:1707.01873..
[13] D. Larimer, Delegated proof-of-stake (dpos), Bitshare whitepaper..
[14] M. Du, X. Ma, Z. Zhang, X. Wang, Q. Chen, A review on consensus algorithm of blockchain, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2017, pp. 2567–2572..
[15] M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery, ACM Transactions on Computer Systems (TOCS) 20 (4) (2002) 398–461.
[16] NEO White Paper, http://docs.neo.org/en-us/, accessed 2019, 2014..
[17] The ThunderChain, https://blockchain.xunlei.com/en/site/index.html, accessed 2019, 2017.
[18] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, M. Dahlin, Separating agreement from execution for Byzantine fault tolerant services, in: The Nineteenth ACM Symposium on Operating Systems Principles, 2003, ACM, 2003, pp. 253–267..
[19] R. Kotla, M. Dahlin, High throughput Byzantine fault tolerance, in: International Conference on Dependable Systems and Networks, 2004, IEEE, 2004, pp. 575–584..
[20] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva: Speculative Byzantine fault tolerance, in: Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles, 2007, pp. 45–58..
[21] Y. Amir, B. Coan, J. Kirsch, J. Lane, Byzantine replication under attack, in: 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), IEEE, 2008, pp. 197–206.
[22] A. Clement, E.L. Wong, L. Alvisi, M. Dahlin, M. Marchetti, Making Byzantine fault tolerant systems tolerate Byzantine faults, in: 6th USENIX Symposium on Networked Systems Design and Implementation, vol. 9, 2009, pp. 153–168..

[23] T. Wood, R. Singh, A. Venkataramani, P. Shenoy, E. Cecchet, ZZ and the art of practical BFT execution, in: Proceedings of the Sixth Conference on Computer Systems, 2011, pp. 123–138.
[24] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand, Scaling byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 51–68.
[25] J. Chen, S. Micali, Algorandr, arXiv preprint arXiv:1607.01341, accessed 2019, 2017..
[26] Y. Wu, P. Song, F. Wang, Hybrid consensus algorithm optimization: a mathematical method based on POS and PBFT and its application in blockchain, Mathematical Problems in Engineering (2020).
[27] S.M. Shafi Goldwasser, C. Rackoff, The knowledge complexity of interactive proof systems, SIAM Journal on Computing 18 (1) (1989) 186–208.
[28] S. Goldwasser, S. Micali, Probabilistic encryption, Journal of Computer and System Sciences 28 (2) (1984) 270–299.
[29] K. Kurosawa, O. Watanabe, Computational and statistical indistinguishabilities, in: International Symposium on Algorithms and Computation, Springer, 1992, pp. 430–438..
[30] J. Katz, Y. Lindell, Introduction to Modern Cryptography, Chapman and Hall/CRC, 2014.
[31] W. Mao, Modern Cryptography: Theory and Practice, Pearson Education India, 2003.
[32] F.P. 180-2, SHA256 Standard, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, accessed 2019, 2001..