J Inf Process Syst, Vol.14, No.1, pp.101~128, February 2018
https://doi.org/10.3745/JIPS.01.0024

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Survey about Consensus Algorithms Used in Blockchain

Giang-Truong Nguyen* and Kyungbaek Kim*

Abstract

Thanks to its potential in many applications, Blockchain has recently been nominated as one of the technologies exciting intense attention. Blockchain has solved the problem of changing the original low-trust centralized ledger held by a single third-party, to a high-trust decentralized form held by different entities, or in other words, verifying nodes. The key contribution of the work of Blockchain is the consensus algorithm, which decides how agreement is made to append a new block between all nodes in the verifying network. Blockchain algorithms can be categorized into two main groups. The first group is proof-based consensus, which requires the nodes joining the verifying network to show that they are more qualified than the others to do the appending work. The second group is voting-based consensus, which requires nodes in the network to exchange their results of verifying a new block or transaction, before making the final decision. In this paper, we present a review of the Blockchain consensus algorithms that have been researched and that are being applied in some well-known applications at this time.

Keywords

Blockchain, Consensus Algorithm

1. Introduction

First introduced by Haber and Stornetta [1], Blockchain is recently considered as one of the technologies with the most potential. It has attracted intense attention only after the introduction of Bitcoin by Nakamoto [2]. This is because Bitcoin has the ability to solve a problem in the traditional payment method: that of trust in a single third party. Normally, when people make payments, they have to trust a third party, who would verify the validity of their transactions, before putting them into action. Unfortunately, this middle party is doubted, whether it could be exploited to cheat its users or not [3]. The problem comes from formal centralization, in which everything depends on a single organization, causing trust to be insufficient. The solution to this problem is using multiple independent organizations to verify transactions, which changes the view from centralization to decentralization.

Motivated by this idea, there is a ledger in Bitcoin and other later variants of Blockchain that records every transaction which has been successfully verified. For example, Bob sends Mary 5 dollars, Mary

Manuscript received November 17, 2017; first revision December 27, 2017; accepted January 3, 2017.
Corresponding Author: Kyungbaek Kim (kyungbaekkim@jnu.ac.kr)
* School of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea (truongnguyengiang.bk@gmail.com, kyungbaekkim@jnu.ac.kr)

sends Carl 10 dollars. So, based on this ledger, it is possible to know how much money a person owns. In order to make the system trustful by the decentralization method, this ledger is held by many organizations, which can be termed nodes or parties. Satoshi has proposed a design of this ledger that introduces a so-called block, which contains successfully verified transactions. The mentioned ledger above is re-established, and contains many blocks with transactions inside. This is the reason for the name "Blockchain". The first block in the chain is called the genesis block [4], which contains the first transactions of the Bitcoin.

When any transaction is proposed, its validity is verified by some nodes. If it is valid, which means the sender has enough money to send (proved by the past transactions in the old blocks), and also the sender has confirmed this transaction by signing his or her digital signature [5] inside the transaction, it will be included in a block. From here, in order to make a transaction really valid, the block containing this transaction should be added to the chain, which must be recognized by all the other nodes. A node will try to append a block containing many transactions by broadcasting it to the rest, which proposes to them to add this one to their current chain. However, if the transactions are verified in this way, there could be confusion when every node tries to broadcast their found block. In order to prevent this situation, an agreement, which is called consensus algorithm, should be made between all nodes about which blocks should be appended, and which nodes are permitted to append their proposed blocks. To date, many consensus algorithms have been proposed.

Many variants of Bitcoin have been introduced since 2009, like Ethereum [6], and Nxtcoin [7,8]. The common characteristic of these variants is that anyone wanting to maintain the ledger can join, as well as withdraw anytime. Therefore, these kinds are called Public Blockchains. Because of this joining freedom, the idea of applying human real-life consensus [9], in which the results should be collected from all the nodes before making the final decision, is difficult. The reason is the unknown number of nodes, and their un-manageable execution. When there are too many nodes, the communication to exchange agreements between them would be very complicated. Worse, because it is possible to manage those nodes, taking agreement from all of them would lead to many possible risks, which could potentially harm the ledger holding procedure. Consequently, in order to append a block to the chain, all the nodes have to show they are more "qualified" than the others to do this work. Therefore, a consensus algorithm of this kind could be named a "proof-based consensus". In public Blockchain, in order to encourage nodes to maintain the ledger, the node will receive some rewards, after appending a block to the chain.

Motivated by the former research of Back [10], the first variant of a proof-based consensus algorithm is proof of work (PoW) [2]. In this kind of consensus algorithm, nodes are only permitted to broadcast their blocks when they have performed a lot of effort by their computing power. Based on the drawback of PoW [11], proof of stake (PoS) employs the stake of each node, and a lucky factor to decide the block appending one. At the time of writing, PoW- and PoS-based consensus algorithms are the most popular ones used in research and applications. Besides these two, there are also other proposed algorithms, like proof of elapsed time [12], proof of luck [13], and proof of space [14].

Realizing the potentiality of Blockchain, many giant enterprises and organizations, like IBM and JP Morgan, have started to research this technology [15,16]. Based on the main rule that only permitted nodes could join the verifying system, many new platforms of Blockchain have been created, which include consortium Blockchains and private Blockchains. Specifically, while the former includes

verifiers from different consortia, only a single organization is included in the latter. Thanks to the limited number of manageable nodes, these variants could employ a consensus algorithm that simulates a real-life situation: the final decision is made based on the results of the majority. This is the basis for giving these variants the name "voting-based consensus algorithms". If a node wants to append a block to its chain, a requirement must be satisfied: this node will have to make sure that at least T nodes (T is a threshold, which varies depending on the system, and T should be more than 50% of all the nodes) will append the same block to it. In order to avoid bad situations that could harm the final results, some fault tolerance techniques [17,18] should be applied. Nodes executing the voting based consensus algorithm follow two popular systems: crush fault tolerance [17], and Byzantine fault tolerance [19].

From here, we wish to emphasize that the proof-based consensus algorithm is not only applicable to public Blockchains, but also to private and consortium ones. The reverse case is also true with voting-based consensus in private and consortium Blockchains. However, the proof-based consensus algorithms are suitable for applying in the network having a lot of nodes. Meanwhile with only a limited number of nodes, employing a real-life solution with voting-based consensus algorithms should be preferable.

In this paper, we present a survey of many variants of consensus algorithm inside Blockchain, which could be categorized into two main types. The first type is the proof-based consensus algorithm, which is often used in public Blockchain. The second type is the voting-based consensus algorithm, which is often used in private and consortium Blockchain. The rest of this paper is organized as follows: Section 2 overviews some of the main entities inside Blockchain. Section 3 presents the main content of the paper about different consensus algorithms, which have been researched and proposed in Blockchain. Section 4 discusses the proof-based and voting-based consensus algorithms. Finally, Section 5 concludes the paper.

2. Overview of Blockchain

This section provides an overview of Blockchain, which includes the transactions verification and block architecture inside Blockchain.

2.1 Transactions Verification

The first work that every system has to do is to verify the identity of the sender, to make sure of one thing: the transaction between the sender and the receiver is requested by the sender, and not by anyone else. For example, when Bob sends Alice 10 dollars, then the request of this transaction comes to a third-party verifier; this middleman has to make sure that this message does indeed come from Bob.

Fig. 1 shows an overview of this verification. In order to do this work, two definitions are employed: public and private keys [20], which are known as digital signatures. When any user makes a transaction, he or she has to use his or her private key to "sign" the transaction, which could be understood that the transaction and his or her private key are used as inputs to a sign function to create a signature. In Bitcoin, the algorithm used for creating a signature for each transaction is the elliptic curve digital signature algorithm [21]. Then the transaction combines with this signature to become a request

transaction. The verifier will check if this entity belongs to the correct person or not, by using the sender's public key, which is known by everybody. Afterward, the transaction, sender's signature, and his or her public key will be inputted together to a verify function to get the result: true or false. If the result is true, then the requester is the true sender in the transaction, and vice versa. Because the signature in each transaction contains 256 bits, if anyone wants to fake this signature to make a fraudulent transaction, he or she has to guess 2^{256} cases, which is impossible.

Besides checking the validity of the sender, the verifier also has to check the validity of the transaction as to whether the sender has enough money to send to the receiver, or not. This work could be done by looking at the ledger, which holds information about every past successful transaction.

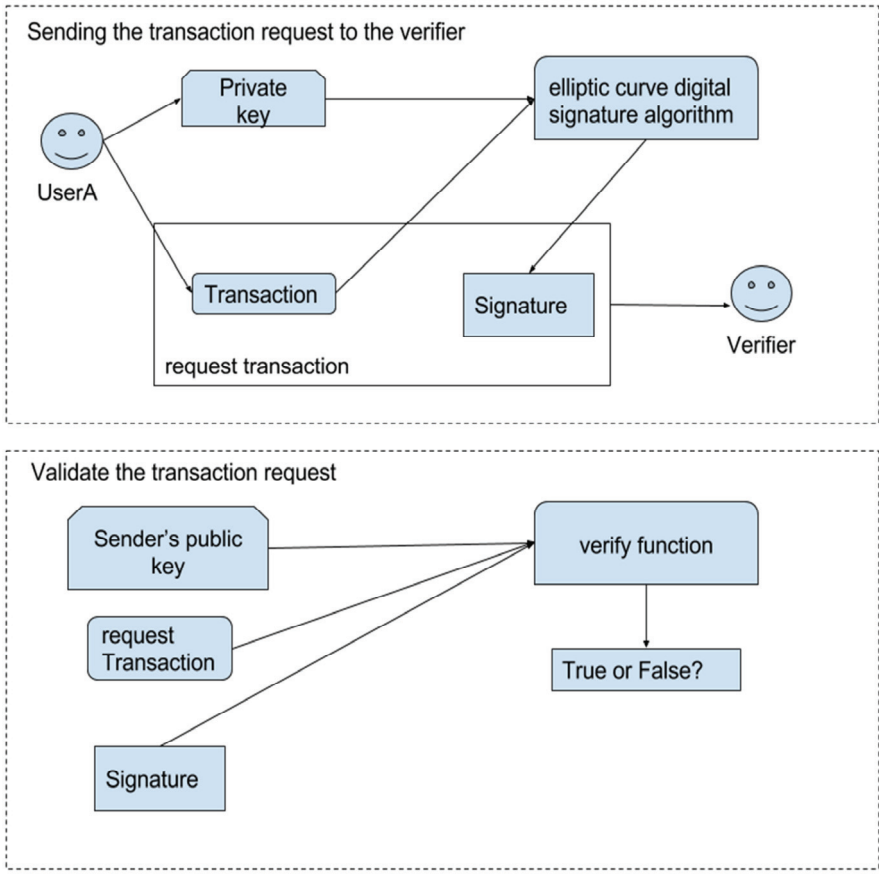

Fig. 1. How a verifying system checks if the transaction is requested by the true sender.

2.2 Blockchain Architecture

As mentioned before, the key phenomenon giving rise to the name Blockchain is the series of blocks, which connect sequentially to each other like a chain. Each block contains many transactions, which are validated, as shown in the previous section. Fig. 2 describes an example of the Blockchain. In Fig. 2, only some main entities inside the block header are described.

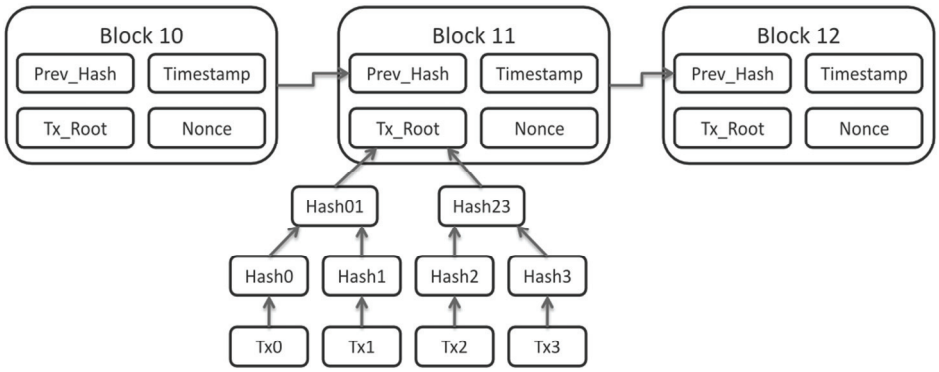

Fig. 2. An example of Blockchain [22].

Specifically, Fig. 3 describes the architecture inside a block. Besides the list of transactions contained inside the block, the block contains some fields in the block header:

- *Prev_Hash*: this field can be known as a reference to parents, which is a link of a block to its previous one in the chain. All the information inside the previous block will be inputted to a hash function to get a value, then this value will be assigned to the field *Prev_Hash* in the new block. In Bitcoin, a 256-bit hash function is used to get this value [23].
- *Timestamp*: the time when the block was found.
- *Tx_Root*: this field, which is also known as the Merkle root [24], contains the hash value of all validated transactions of the block. As seen from the example in Fig. 2, all the transactions are hashed into a hash value; then they combine with each other pair-by-pair, and are inputted to another hash function. This work is repeated, until there is only a single entity, which stands for the Merkle root.
- *Version*: this field contains the protocol version used by the node proposing the block to the chain.
- *Nonce*: this field is used in PoW, which proves the efforts that a node has paid for getting the right to append his block to the chain. This field will be presented in the next section.
- *Bits*: this field indicates the difficulty level of the PoW, which will be introduced in the next section.

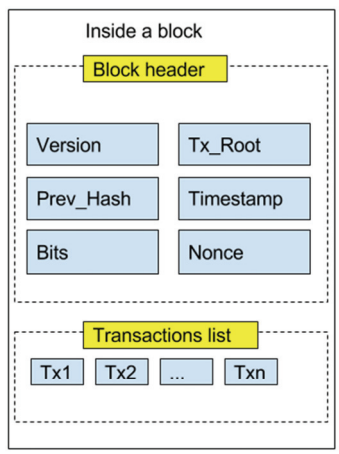

Fig. 3. The fields inside a block.

3. Blockchain Consensus Algorithm

This section will finish some definitions that are offered in the previous sections without much clarification, like the nonce field, bits field in Section 2.2, and PoW which is proposed for appending a new block to the chain. Further, the main focus of this section is on how nodes in the verifying network agree with each other about the ledger that they hold. This section is divided into two main sub-sections: proof-based consensus algorithm and vote-based consensus algorithm.

3.1 Proof-Based Consensus Algorithm

This subsection introduces the proof-based consensus algorithm. The original work is PoW, proposed by Nakamoto [2]. To date, many variants of proof-based consensus algorithms have been proposed, which are based on PoW, PoS, their hybrid form, and other variants that are made independently from these two major ones. The basic concept of proof-based consensus algorithm is that among many nodes joining the network, the node that performs sufficient proof will get the right to append a new block to the chain, and receive the reward.

3.1.1 PoW-based consensus algorithm

3.1.1.1 Original proof of work

As mentioned, in the Blockchain network, if every node tries to broadcast their blocks containing the verified transactions, confusion could possibly arise. For example, consider a transaction that is verified by many nodes, who will then put it into their blocks, and broadcast to other nodes. If the broadcasting work is free, this transaction could be duplicated in different blocks, then the ledger is meaningless. In order to get agreement between all nodes about the newly added block, the PoW requires each node to solve a difficult puzzle with adjusted difficulty, to get the right to append a new block to the current chain. The first node who solves the puzzle will have this right.

Specifically, before solving this puzzle, all the verifying nodes would have to put their verified transactions, as well as other information like *Prev_Hash* and *Timestamp*, into a block. Then they start solving this puzzle, by guessing a secret value, which is the nonce field as introduced in Section 2.2, then put it into the block. All the information inside the block header will be combined together, and inputted to an SHA-256 hash function [25]. If the output of this function is below a given threshold T, which is designated by the difficulty, the secret value is accepted. Otherwise, the node has to make another guess of the secret value, until he gets the answer. The difficulty of the puzzle will be adjusted after every 2016 blocks are appended, so that the average speed for adding a new block in the chain is 1 block per 10 minutes. Also, the more difficult the puzzle is, the smaller the threshold T is. Fig. 4 describes the processing for handling the guessed value. Thanks to the usage of SHA-256, guessing this value is extremely difficult, which makes every node guess many times to get the answer, unless they are lucky enough. Because of the efforts paid for guessing the right value, this work is called the PoW. Also, the node joining the network using PoW can be called a miner, and the action of finding a suitable nonce is called mining.

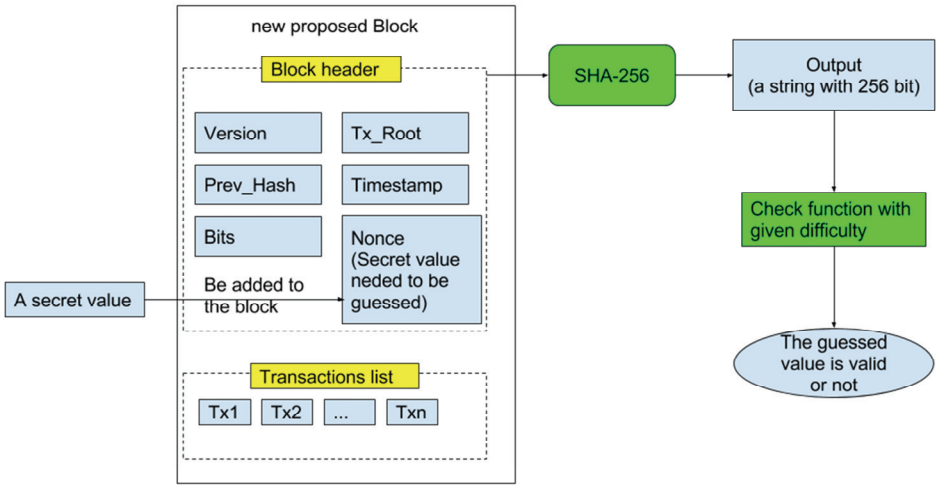

Fig. 4. The process for handling with the nonce: guessed secret value.

When a node finds the secret value, he broadcasts his proposed block with this value to other nodes, to notify them that the answer has been found. Right after that, all the miners receiving this message, who have still not found the secret answer for their puzzles, will stop guessing. Instead, they check the broadcasted block for whether all the transactions are valid; if the block contains the *Prev_Hash* value is the hash value of the last block in their chain, the validity of the nonce field … If all the verifications are correct, then these nodes will append the proposed block to their current chain, and re-start guessing the secret value, by repeating the steps above again.

However, there is a rare case when more than 1 miner finds the answers for the puzzle, before it being noticed that another miner has also found another suitable answer. At that time, these miners will still broadcast their block with the found nonce. Then, other miners who receive the first coming block will ignore the others coming later. This work leads to the forking problem: in the verifying network, there are different chains of blocks (they should be the same). Satoshi proposed that those miners will keep mining a new block on their forks, until one fork is made longer than the others. So at this time, all nodes have to follow this longest fork. Fig. 5 describes the forking problem and the PoW solution to handle it. Whenever a block is recognized in the chain by all the nodes, the miner appending this block will receive some bitcoins as a reward.

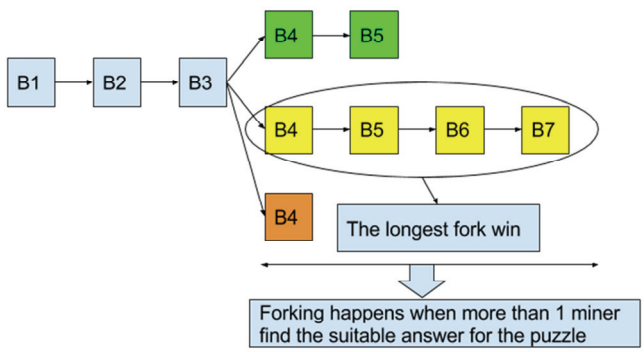

Fig. 5. How Bitcoin handles the forking problem.

3.1.1.2 Variants based on PoW

Unfortunately, there are many works and research efforts that state the drawbacks of PoW, which include some security issues and usage issues. Therefore, many variants are made to mitigate those drawbacks.

Thanks to modern hardware, the block appending speed increases day-by-day, which makes the difficulty of the puzzle harder. Therefore, in order to be the first puzzle solver, the miners have to invest more in the hardware. This investment results in other miners, who do not have good condition, being unable to compete effectively. Tromp [26] have proposed replacing puzzle work with an idea employing the Cuckoo hash function [27], which allows miners to pay fewer efforts, but win the right to append the block easier. The authors' idea is using a hash table with two different hash functions that have all the possible results findable on the hash table. Then, some values are inputted to these hash functions, which results in two positions in the hash table. If one of the two positions is blank, the hashed result will be put into one of them. But if none is blank, then the newly hashed result will remove the old one in the hashed position of the first hash function, and replace it. Afterwards, the old replaced result is inputted to the two hash functions above, and the procedure is repeated as before. This repetition will be done until a definite loop is made, or there is no longer a blank position in the hash table. The work of the Cuckoo hash function can create a graph, in which the vertices are the hashed position or the replaced position, with edges being the values that have been removed. To replace the puzzle work in the original PoW, a miner will have to guess many nonces, which are then inputted to the defined hash functions. When the graph created by these guessed nonces has sufficient cycles, the miner will broadcast his block with his guessed nonces inside.

King [28] proposed the idea that instead of finding a meaningless nonce satisfying the below-threshold requirements, all the miners will have to find the longest chain of primes, which has to satisfy some requirements. The first requirement is its length has to be larger than a given value. Then the second requirement is that it has to be in the form of a Cunningham chain [29]. Finally, it has to have a defined origin value; which is the center of the two first primes in the chain, being divisable by a required number. When a node broadcasts his block, other miners will check this chain by using an algorithm called the Fermat Primality Test [30]. Besides the main aim of verifying the transactions, this kind of PoW variant is also supposed to dedicate to mathematics by finding the distribution of the Cunningham prime chain.

Being proposed by many researches, one of the PoW's drawback is a problem called the Double spending attack [31], whose script is described in Fig. 6. In short, an attacker would try to reverse a transaction that has been verified by some nodes, by making another transaction that could make the first one invalid in another fork, then try his best to make this fork longer than the others. In order to do this work, he or she has to have at least 51% computing power of the verifying network [32]. Therefore, the double spending attack can also be called the 51% attack. It is quite difficult for any individual node to have such computing power.

However, with the appearance of so-called pool mining [34], this kind of attack could be possible. In pool mining, instead of each individual miner trying to guess the nonce value, they gather into a group, then vote for a pool operator and find the nonce together, which makes the work much easier. The rewards would appear in a place called the coin-base, so that afterwards, every miner joining this pool can receive their incentives, which are equal for each of them. Eyal and Sirer [35] have presented the

risk of a 51% attack with this kind of mining group. They proposed a method to prevent this kind of risk: before finding the nonce of the block, every node has to own the private signature of the coin-base. Then they have to sign this signature in their created blocks. This work is considered to be too dangerous, because the miners inside the pool can freely transfer the rewards to someone else [36]. Otherwise, if the private signature is not provided, this group has to solve an extra PoW puzzle with different difficulty, which is much more difficult than a single puzzle. When the block header with a found nonce satisfies the first requirement, the output value of the SHA-256 hash function will be inputted into this function again to satisfy another requirement like before, with different difficulty.

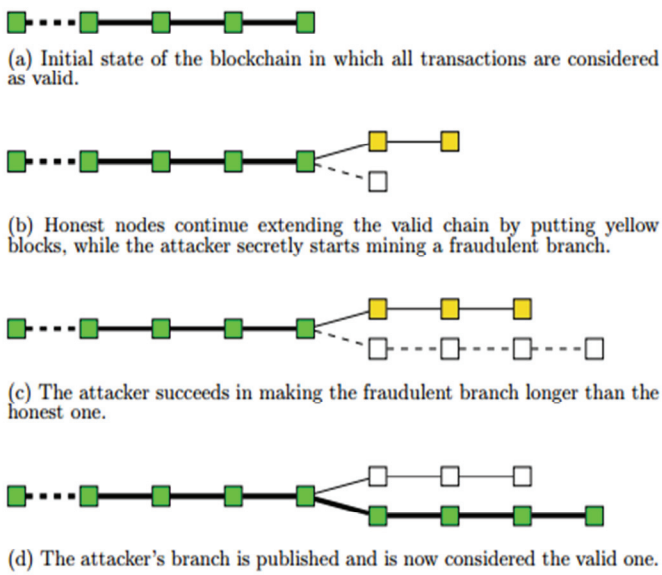

Fig. 6. A script for double spending attack [33].

In order to discourage pool mining, Miller et al. [37] have proposed a mechanism for changing original puzzles into so-called non-outsourceable puzzles, which provide the miner inside any pool with the chance of winning the rewards, without making any effort to solve the puzzle. This kind of puzzle evolves using a signing key, which could be exploited to execute illegal work against the pool.

Besides the security issues, PoW is supposed by Eyal et al. to be unsuitable for real-time payment [38]. Consider a case when a person goes out and buys a cup of coffee, then because of the creating speed of a block, has to wait for around 10 minutes to have his or her coffee, because the sellers need nodes to verify his or her payment. Therefore, the need to reduce the latency for transactions is very necessary. Two possible solutions have previously been proposed: increasing the block size for containing more transactions, and reducing the difficulty for solving the puzzle. While the former takes longer time to propagate the block on the network, the latter reduces the time for creating a block, but could lead to inconsistency forking of the chain and double spending attack. Eyal et al. [38] have proposed a new consensus model called Bitcoin-NG, which separates the traditional block into two different kinds, called key block and micro block. When a miner finds a suitable nonce for his puzzle, he will broadcast his key block to other miners, and become a so-called leader, until another miner finds another suitable nonce. During the time of being leader, this node will publish some micro blocks that contain

information about the transactions. The transactions can be verified quicker, because while waiting for a new leader to create a new key block, there could be many micro blocks created.

Sompolinsky and Zolar [39,40] suggested reducing the time for appending a new block to the chain with a strategy for handling the fork and preventing the double spending attack problem. This so-called GHOST strategy keeps all the forking blocks that have not been chosen for the main chain. Then instead of choosing the longest fork, the one with the most PoW contributing to it will be chosen. Fig. 7 describes this choosing strategy. The figure shows that the proposed GHOST protocol makes the main chain "defeat" the attacker chain, because it contains more blocks, which proves that there is more PoW contributing to it.

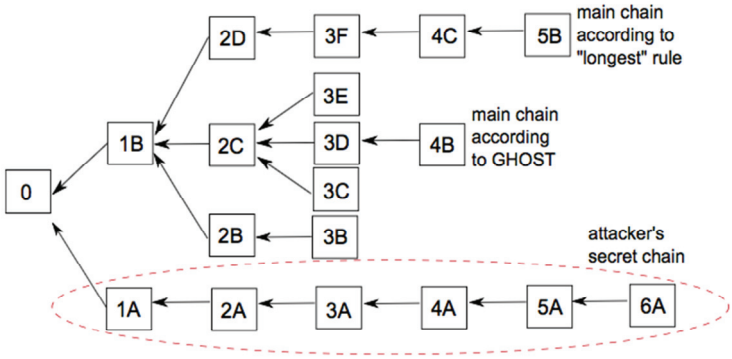

Fig. 7. Strategy for choosing the main chain when fork appears. Adapted from Sompolinsky and Zolar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*. Heidelberg: Springer, 2015, with the permission of Springer [40].

Liu et al. [41] have proposed a generalized Proof of Work consensus algorithm based on PoW. The block is not appended by a single miner, but by a committee of miners with a luck factor. The difficulty in this work is set much easier than the original PoW, which enables the miners to be able to find many possible nonces with their computing power. In each round of appending, each miner will have to find as many nonces as possible for their created block. After any nonce is found, its finder will send the block, including this nonce, to a committee, including some other miners, who have the responsibility to validate the received block, and put the valid block into a temporary array. At the end of the round, a random number is chosen. The miner having the block whose index in the array is equal to this random number is chosen to not only append the block to the chain, but to also be a new member in the committee. After a duration, some members in the committee will be rotated, to make sure that the number of members is not too large. After each round, the rewards will be given equally to all the members inside the committee.

3.1.2 PoS-based consensus

The proposed PoW is supposed to be unfair: while some miners owning modern and powerful equipment could find the suitable nonce easier, others with poorer condition could find it very difficult to be the first one to find a suitable nonce. PoS is supposed to deal with this inequality. Being firstly introduced and discussed in Bitcoin forum from 2011 [42], PoS has had some variants and research contribution to it. The basic idea of these consensus algorithms is using the stake to decide who will get

the chance to mine the next block of the chain. Using stake as a proof has an advantage: anyone who owns much stake would be more trustful. He or she would not want to perform any fraudulent actions to attack the chain that contains much of his or her profits. Furthermore, using PoS would require any attackers to own at least 51% of all stakes in the network to perform a double spending attack, which is very difficult.

There are currently two popular kinds of consensus using PoS: the kind using pure stake to make consensus, and the hybrid kind, which combines PoS and PoW.

3.1.2.1 Pure stake-based consensus

The pure PoS form could be seen in Nextcoin [8] (aka. Nxtcoin, http://nxtcrypto.org/). In this platform, the more stake a miner owns, the more chance he has to mine a new block. Specifically, if there are a total b coins from all the miners, and miner M owns a coins ($a < b$), the chance for miner M getting the right to mine a new block is a/b. The lucky miner picking work is executed every 60 minutes, and this work is made randomly based on the stake of each miner, as mentioned. Once a miner gets the chance to mine a new block, he will verify the transactions, collect them into a block, then broadcast it to the other miners, and receive the rewarding fee.

Being similar to Nextcoin, Bentov et al. [43] proposed a method for finding the miner based on the pure stake he owns: the more stake a miner has, the more chance he will get to become the block appender. However, the miner getting this chance is chosen based on the state of the block. Also they proposed a definition called follow-the-Satoshi procedure, and they considered a Satoshi value, which is the smallest currency unit. Follow-the-Satoshi procedure will get input as a Satoshi index, which is between 0 and the total number of Satoshi, then it will find the block that created this Satoshi, which could be understood as the rewards for a miner to create this block. Afterwards, all the transactions including this satoshi will be found out, to identify the last owner of this satoshi, who will become the one appending the next block. The work of choosing the Satoshi is based on a hash function, which takes some inputs based on the current status of the chain. The first input parameter is a value that is gotten from a so-called *comb* function owning inputs as some bits. Each of them is the first bit gotten by hashing each existed block in the chain sequentially. The second input is taken from the number of current blocks in the chain, and the third input is a random integer. Bentov et al. [43] also proposed a solution to handle the problem, in which the owner of a chosen Satoshi does not create a block when he has the chance. If a chosen satoshi after 3 consecutive chosen times could not experience a new created block, it will be blacklisted. Function *comb* is chosen to satisfy that it would be difficult to influence the choosing of *satoshi*, which has been issued by Russell and Zuckerman [44] with collective sampling, and Ben-Or and Linial [45] with leader election and collective coin-flipping.

Kiayias et al. [46,47] also employed the idea of Bentov, follow-the-Satoshi procedure to execute the PoS consensus. They claim that the leader election should be done randomly by calculating an entropy value. This calculation should be secure enough, so that it would be difficult to simulate the protocol to predict the calculation, and manipulate the leader election. Kiayias et al. considered the work of Bentov et al. [43] for leader election as a way to calculate entropy based on the current status of the Blockchain. The first difference of Kiayias et al.'s work from Bentov et al.'s is that they snapshot the stake of each stake holder after an interval called an epoch. In each epoch, a collection of stake holders will be chosen to execute the so-called coin-flipping protocol. Based on these results, the leaders and the stake holders

executing this protocol in the next epoch will be chosen, which makes sure that the entropy calculation is difficult to simulate. Also, in order to raise the equality, the leader will have to create the block only, while the work of adding transactions belongs to a group of stake holders called endorsers, who are voted like the leader. The rewards will be divided equally to those leader and endorsers.

Using the stake as a proof for voting, not for getting the chance to mine a new block, is the idea of Larimer [48]. This kind of consensus algorithm is called delegated proof of stake. In this platform, there are many people who hold stake, and they will have to vote for a delegation, which includes some "witnesses", who are miners verifying the transactions and maintaining the chain. The more stake a person owns, the more powerful voting he has to assign the witness. After the verifying congress is made, the witnesses inside will verify the transactions, and produce blocks, including the valid ones. The list of witnesses is always shuffled. With the creating speed of 2 seconds per block, the witness in the list will have to produce blocks sequentially. If any witness fails to produce his block, he would potentially be removed from the delegation. Whenever a witness creates a block to append to the chain, he will receive a reward.

3.1.2.2 Hybrid form of PoW and PoS

Being officially introduced in a paper by King and Nadal [49], PPcoin could be considered as the first variant of PoS, but it still uses PoW. These authors have proposed a definition called the 'coin age' of each miner, which is calculated by his stake multiplied by the time that the miner has owned it. For example, Bob receives 10 coins from Alice and keeps them for 10 days; consequently, he has 100 coin-day, and if he sends 2 coins to anyone, the coin age of these 2 coins accumulated will be destroyed. In order to get the right to append a new block to the chain, the miner creates a kind of block called a coin stake, which like before, holds many transactions, but includes a special one from that miner to himself. The amount of money spent on this transaction will provide the miner more chance to mine a new block. Afterwards, he will have to do a puzzle, like PoW. The more money he spends on the transaction, the easier the puzzle he has to solve. If any miner solves the puzzle first, he will get 1% of the amount of coins he has spent in the transaction, but the coin age accumulated by these coins will be reset to 0.

Being dissimilar to King and Nadal [49], Vasin [50] do not employ coin age with their Blackcoin, because they suppose that using coin age could provide the attacker the chance to accumulate enough value to cheat the network. Worse yet, there could be some miners who hold their stake until they have a lot of coin age, while staying offline from the verifying system. Therefore, Vasin [50] propose using the raw stake instead of coin age for providing miners the chance to mine a new block, which could encourage more nodes to be online to get the rewards. Being noticeable with the offline miner, Ren [51] propose using an exponential decay function with the coin age: the more the miner waits for the increase of the coin age, the less speed of increase it has.

The problem of double spending attack is again shown by Duong et al. [52]. The problem with miners owning more than 51% mining power raises a high alert for the security of Blockchain. They proposed a method for mitigating the double spending attack by combining PoW and PoS. The aim of this action is to make sure that even if a miner owns more than 51% of the mining power, he still does not have much chance to make a fraudulent action. These authors propose using the PoW first for choosing a winner, who is the first one getting the answer of the puzzle. Subsequently, this winner will not only append a block called PoW block to the chain as usual, but he will also provide a basis to

choose another miner who owns stake. This basis is: if the return value of a hash function, whose input parameters are the newly appended PoW block and stake owner's private key, is below a threshold, the chosen miner will have the chance to append a so-called PoS block to the chain. Duong et al. [52] claim that each PoS block is linked to a single PoW block, and each PoW block is linked to a previous PoS block. Consequently, it is difficult to make a double spending attack. The reason is that assuming that an attacker could own more than 51% of the computing power, then he could get the right to be the miner appending the illegal PoW block to the chain. However, right after that, there is still a PoS block appended by another miner chosen by the mentioned base. This stake miner will see the longest chain at that time, and realize that the latest one is a PoS block, which he could not append his PoS block to. As a result, the attack fails. The double spending attack is supposed to happen only when the attacker owns not only more than 51% of the computing power, but also more than 50% of all the stake holders. This kind of consensus is not based on the amount of stake, but only for miners owning stake.

As an improvement of [52], Chepurnoy et al. [53] have pointed out a problem with their former research. Firstly, the executing environment is static, because at each round, the invested physical hardware for mining the resources and virtual resources known as stake remain the same; which assumption does not equate with reality. Also, the usage of stake has not been employed, which would not be fair for the stakeholders who own more stake than others. The authors suggest that there should be a difficulty adjustment for both the PoW and PoS when the environment changes. Therefore, Chepurnoy et al. [53] propose adjusting the difficulty based on the environment—the rate of block creation. Specifically, their proposed difficulty for proof of work follows the original Bitcoin's one, while the PoS difficulty is inverse to the stake that the stake owner has. For example, if a miner owning 1 coin has the hash value mentioned from [52] below threshold t, then another miner owning b coin will only have to satisfy the hash value below $t * b$.

Bentov et al. [54] propose a solution called Proof of Activity. This combines PoW and PoS in order to not only solve the double-spending attack, but also handle some tragedies made by PoW called the tragedies of the common. The first mentioned tragedy is that only the miners solving the puzzle get the reward, while the others who have the role of preserving the ledger, update it and validate the new block; they do not receive anything. The second one is that miners can co-operate with others to raise the transaction fee to be high to charge the users. This action could make the Blockchain useless, because nobody would want to use it. In order to deal with these tragedies, these authors proposed creating an empty block by PoW first: all the miners will try to find a nonce with a block without any transactions. After finding a suitable nonce, the miner will broadcast the block containing it to other miners, who will verify the validation of their received ones. Also, they will check if they are the winner of another lottery, which is designed based on the block they receive. This lucky chance is like follow-the-Satoshi procedure in [43]. A hash function is suggested to be used N times, which input parameters are the hash value of the received block, the hash of the latest appended block, and a random number, which is randomized N times. N miners who see that they own the N chosen Satoshi will continue the work: the $(N - 1)$ first miners will sign into this empty block, their signature proving that they own the chosen Satoshi, then broadcast this signature. The last miner will include not only the required signature, but also as many transactions as he wants, then broadcast the block to other miners. The block creater and N chosen miners will get equal rewards. Besides preventing a double spending attack, this kind of consensus also encourages miners to stay online to collect the rewards.

3.1.3 Comparisons between PoW, PoS and their hybrid forms

Table 1 shows the comparisons between PoW, PoS and their hybrid form. As can be seen, using electric power to guess the secret value executing PoW, a lot of money should be spent on not only hardware devices, but also on the energy to execute them. In contrast, PoS does not employ any puzzle, so these two investments are much lower.

Table 1. Comparisons between PoW, PoS, and their Hybrid form

Criteria	PoW	PoS	Hybrid form of PoW and PoS
Energy efficiency	No	Yes	No
Modern hardware	Very important	No need	Important
Forking	When two nodes find the suitable nonce at the same time	Very difficult	Probably
Double spending attack	Yes	Difficult	Yes, but less serious than in PoW
Block creating speed	Low, depends on variant	Fast	Low, depends on variant
Pool mining	Yes, but it can be prevented	Yes, and it is difficult to prevent	Yes
Example	Bitcoin	Nextcoin	PPcoin, Blackcoin

In PoW, forking can happen if two miners find a suitable nonce at the same time. Meanwhile with PoS, it is very difficult, happening only when a miner can own up to 51% of all stake in the whole verifying network [49]. By employing both PoW and PoS, although their hybrid form can still make a fork, the probability of causing a double spending attack is much lower, compared to the pure PoW.

Considering the speed of creating a new block in the chain, overall almost all variants of PoW require much time to append a new block to the chain, while with PoS, the block creating speed is lower, because no node has to solve any puzzle.

What could make PoS less attractive is the work with pool mining: for PoW, the miners joining pool are trackable, and the fraudulent work is preventable, as in [35,37]. But in PoS, it would be very difficult to prevent the miners inside a pool delegating their stake to a single miner as pool operator.

3.1.4 Other kinds of proof-based consensus algorithm

Blocki and Zhou [55] pointed out the same problem as King [28]: The PoW wastes too much energy in finding the nonce, and also it is meaningless in everyday human life. Worse yet, it is not fair for the miners having poorer condition to purchase modern hardware, which makes their chance of mining a new block very low. Therefore, Blocki and Zhou [55] proposed using some kinds of puzzle for education and social activities, which would be easy for computers to solve, but difficult for human to solve. It is for the human to solve the puzzle to mine a new block, instead of using hardware. This difference would be fair for everybody, whether they could or could not invest in new modern equipment.

Proof of burn [56] and proof of space [14,57] are other kinds of proof-based consensus algorithms that do not employ the idea of PoW and PoS. In proof of burn, miners have to send their coins to an address to "burn" them, which means these coins could not be used by anyone else. The miner who burns the largest amount of coin during a duration will be the one getting the right to mine a new block. With Proof of Space, miners will have to invest their money on hard disk, which is much cheaper than computing devices for PoW. During the work, the proof of space algorithm will generate many large datasets called plots on the hard dish. The more plots a node has, the more chance he will get to mine a new block.

Proposed by Intel [58], proof of elapsed time [12] is used in a blockchain platform called Sawtooth Lake [59]. This kind of consensus is executed in a special environment called the trusted execution environment (TEE) [60], with a special device of Intel known as Intel Software Guard Extensions (XGS) [61]. In order to conduct the consensus algorithm, each miner will at the same time request a wait-time from a trusted enclave, which is also known as a trusted function inside XGS hardware. Subsequently, all the miners will receive their responded wait-time from the enclave, and together will wait until their received time elapses. When a miner waits enough time, but has found no one has finished the waiting match, he will broadcast to all other miners that he is the winner, which provides him a chance to mine new block. In short, the miner owning the shortest received wait-time will be the one to mine a new block. In order to support this kind of consensus, two functions are used: function CreateTime will inform the miners of the time they need to wait, and function CheckTimer will check whether or not the miner has waited enough time. Because this consensus is executed in TEE provided by XGS devices, it is supposed that cheating with the work of the two above functions is very difficult.

Milutinovic et al. [13] proposed a kind of consensus, which is also executed in TEE and with XGS devices, called proof of luck. To execute it, after all the ledgers from all the miners are synchronized, each miner will create for themselves a new block to append to their current chain, then a random number ranged from 0 to 1 will be assigned for each created block, which could be considered a lucky value. All of the nodes will have to agree that the chain with the total largest lucky value would be the main chain. As a result, proof of luck is considered to be fair for all the miners. Furthermore, it would be very difficult to make attacks, like double spending attack, because the attacker should be very lucky to perform their illegal actions successfully.

Multichain [62,63] is a kind of consensus algorithm, which is quite similar to PoW with the work of mining and fork handling. However, multichain does not apply PoW for choosing the node to append block to the chain; instead, it applies a round-robin schedule to make nodes create blocks in rotation. Specifically, a parameter p ($0 < p < 1$) called the mining diversity is used. In each phase, all the nodes have to wait for a duration, then start checking their rights to append new block to the chain. If among p^*N (N is the number of participating nodes) blocks that are created most recently, no block is created by a supposed node A, then node A could append his proposed block to his current chain, then broadcast this block to other miners. In the case that fork happens, like PoW, the longest one would be chosen.

3.2 Voting Based Consensus

In order to execute the voting based consensus algorithm, the nodes inside the verifying network should be known and adjustable, so that they can exchange the message easier. This is the main

difference compared to proof-based consensus algorithms, which nodes are often free to join and withdraw from the verifying network. Also, in voting-based consensus algorithm, besides maintaining the ledger, all the nodes in the network would have to verify together the transactions or blocks. They will communicate with others, before deciding to append their proposed blocks to their chain or not. In almost all of these variants, nodes are required to see at least T nodes having the same proposed block with them to do the appending work (T is a given threshold).

Executing voting-based consensus algorithms is very similar to traditional methods for tolerating faults used in the distributed system [64]. Therefore, voting based consensus should be designed to resist some bad cases:

- Some nodes are crashed.
- Some nodes are not only crashed, but also subverted.

In crashing cases, nodes will wait for the messages from other nodes. However, there are some nodes that do not run, then the normal nodes do not receive enough evidence to make the decision. Therefore, in order to prevent the crashing case with f nodes, there should be at least $f + 1$ nodes operating normally [65].

In subverting cases, nodes could act strangely, which could make the results inaccurate. This problem can be presented by a classical problem called Byzantine generals proposed by Lamport et al. [66]: a group of Byzantine generals attacked an enemy's camp. They decided to divide their army into N groups led by N generals, which would attack the enemy from different sites. If they attacked at the same time, they would win; otherwise, they would lose. Consequently, they had to make an agreement with each other about the attacking time by exchanging messages, and following the decision of the majority. Unfortunately, there were some traitors inside the general group, and they wanted to cheat other generals by telling different decisions to the others. Therefore, the results could be made inaccurate, which made some generals attack, while others did not, leading to failure. Lamport et al. [66] have proved that in order to tolerate f subverted generals, there should be at least another $2f + 1$ normal generals to accompany them.

Coming back to the Blockchain, when each node executes the consensus work, some nodes can be subverted, which sends different results to other nodes. Then the worst case is the network could not resist them, causing the ledger to be different in different nodes. Considering the crashing cases, if nodes are crashed, then they could not send their results to other nodes, which makes it difficult to make the final decision.

Consequently, based on these bad situations, the voting based consensus algorithms could be classified into two main kinds:

- Byzantine fault tolerance based consensus: a kind of consensus that could prevent the cases of crashing nodes and subverted nodes.
- Crash fault tolerance based consensus: a kind of consensus that could only prevent the cases of crashing nodes.

All consensus algorithms in these two sub-categories will have to make a trust assumption: among N nodes, there should be at least t nodes ($t < N$) operating normally. While in crash fault tolerance-based consensus, t is usually set equal to $[N/2 + 1]$, in Byzantine fault tolerance-based consensus, t is usually assigned equal to $[2N/3+1]$.

3.2.1 Byzantine fault tolerance based consensus

The famous Hyperledger Blockchain platform [67] has been used by many enterprises. IBM is one of them, with Hyperledger Fabric [68]. A kind of Byzantine fault tolerance called Practical Byzantine Fault Tolerance (PBFT) proposed by Castro and Liskov [18] was used for Hyperledger Fabric [15,69]. In PBFT, there are two kinds of nodes: A leader node, and some validating peers (nodes); and these peers will execute some rounds for appending a block to the chain, which is described in Fig. 8 [69]. Initially, the clients send their requests for transactions to their corresponding validating peers. From here, the receiving peer will validate the transactions, then broadcast them to other peers, including the leader. After the number of transactions reaches a threshold called batch size, or after an interval, the leader node will order the transactions by their created time, putting them into a block. Afterwards, three phases of PBFT are executed. Firstly, in the Pre-prepare phase, the leader broadcasts his proposed block to other peers. They will receive and store the block locally. Then in order to make sure that the received block from the leader is the same, they make a double-check by broadcasting it in the Prepare phase and Commit phase. After the Prepare phase, if any node receives the blocks, which are the same as what they have stored locally before, from more than 2/3 of all the nodes, they will execute the Commit phase. Then the same procedure is recorded after the Commit phase, which is the requirement for any node to execute the transactions in the proposed block, and append it to their current chains.

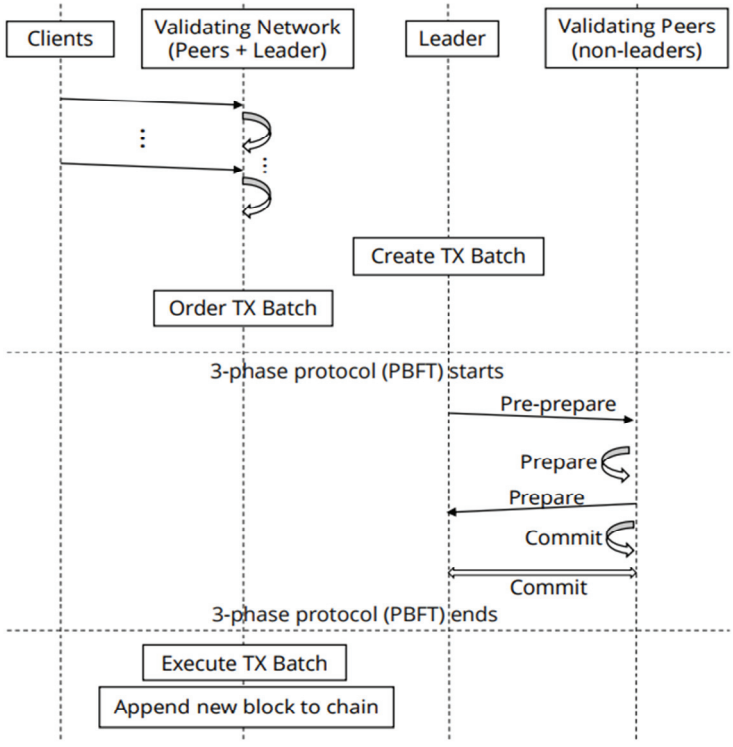

Fig. 8. Rounds for verifying transactions in Hyperledger fabric. Adapted from Sukhwani et al., "Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric)," in *Proceedings of the IEEE 36th Symposium on Reliable Distributed Systems*, Hong Kong, China, 2017, with the permission of IEEE [69].

Another two popular Blockchain platforms, which use Byzantine fault tolerance-based consensus algorithms, are Symbiont [70] and R3 Corda [71]. They employ a consensus algorithm called BFT-SMaRt proposed by Bessani et al. [72]. Fig. 9 describes this algorithm. Basically, this algorithm is similar to PBFT, but uses different names (Propose–Write–Accept vs. Pre-prepare–Prepare–Commit in PBFT). Besides executing the consensus work, Bessani et al. [72] also propose a mechanism for storing the log of executed operations in a single machine, which is used for gaining the last current state, when a node fails, and needs to restart.

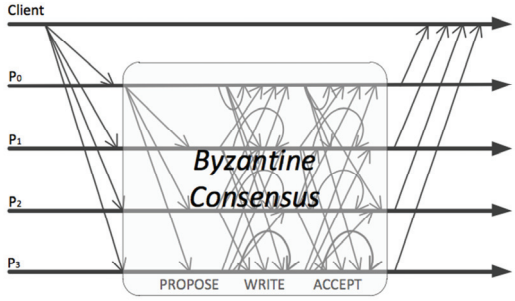

Fig. 9. BFT-SMaRt execution. Adapted from Bessani et al., "State machine replication for the masses with BFT-SMART," in *Proceedings of 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, 2014, with the permission of IEEE [72].

Iroha [73] is another Hyperledger based Blockchain platform, which has the consensus algorithm called Sumeragi [74], being "heavily inspired" by Bchain by Duan et al. [75]. With Bchain, all the nodes will be arranged linearly, so that a given node will only receive message from his previous node, and send messages to his next node in the chain. Because the broadcasting work can be avoided, the load among the nodes is balanced, but with the higher cost of time execution and time for reconfiguring if faults happen. Being similar to PBFT and BFT-SMaRt, implementing Bchain requires at minimum [2*N*/3+1] nodes operating normally.

Ripple [76-78] is a Blockchain platform whose consensus algorithm is made by the new Blockchain comers. Consequently, Ripple consensus algorithm is supposed to improve on the weakness of Bitcoin. In Ripple, the chain of blocks is not used, but rather a raw ledger is employed; and after some verified rounds, transactions will be added directly to the ledger. Like the original Bitcoin, this ledger is maintained by some nodes, which run Ripple Server Software. Ripple adds new transactions to the ledger if they are verified successfully by at least 80% of all servers. To handle the verification of requested transactions, Ripple ledger has two main forms: last-closed ledger, which indicates that all the transactions in it are verified successfully by sufficient servers (80%); and open ledger, which has transactions verified by insufficient servers. To make the verification, instead of broadcasting the transactions to others, each server has their own list, called a unique node list (UNL), which includes some other servers. Following Ripple, the intersection of UNL by any two different servers should be at least 1/5 of all the servers over the network. These servers will only communicate with others in their UNL. Initially, in the consensus algorithm, there would be many transactions created by many clients broadcasted to some servers. Then they would validate each of the received ones, putting the correct ones temporarily in their ledger to change its status to open-ledger. Also, they aggregate all the correct transactions into a so-called candidate set. At this time, there would be some rounds to make an open-

ledger to become a last-closed-ledger. In the first round, each server would aggregate all the candidate sets from other servers in his UNL into his candidate set, then verify the transactions inside this set. A "yes" vote will be put to each of the transactions if they are verified successfully. Subsequently, in the next following rounds, any transactions that do not receive enough "yes" votes will be discarded from the candidate set (the final rounds require at least 80% of yes votes for each transaction). David et al. [59] have stated that in order to maintain the correctness, there should be only a maximum $(n-1)/5$ subverted nodes among n nodes in the network.

As a variant of Ripple, Stellar [79] uses a group, which is similar to UNL in Ripple, called quorum slice for communicating between verifying nodes (validator nodes in Ripple). The validator nodes can belong to one or many different quorum slices. For a given transaction, if a node wants to confirm a given transaction to be valid, he will have to ask the other nodes in his quorum slice. If the transaction is verified successfully by all nodes in his quorum slice, he will consider this transaction to be valid. Fig. 10 describes an example of quorum slices in Ripple, which are organized by hierarchical structure, similar to the example in [79]. Considering "Node 1" in the example, it makes a quorum slice with two other different nodes in "Group 1", and another node among all nodes from "Group 2". Specifically, Nodes (1,2,3,5) and Nodes (1,2,4,6) are two of the quorum slices among many in this example. Also, Group 1 and Group 2 are the top levels; transactions have to be verified first from these two levels. It could be imagined that Group 1 contains some banks, while Group 2 consists of some verification services, and Group 3 is the set of users using the payment service. In order for Node 10 to verify a given transaction, he would have to ask other members in his quorum slice, which could be Node 1, Node 2, and Node 7. If all of them agree with this transaction, Node 10 could consider it to be valid.

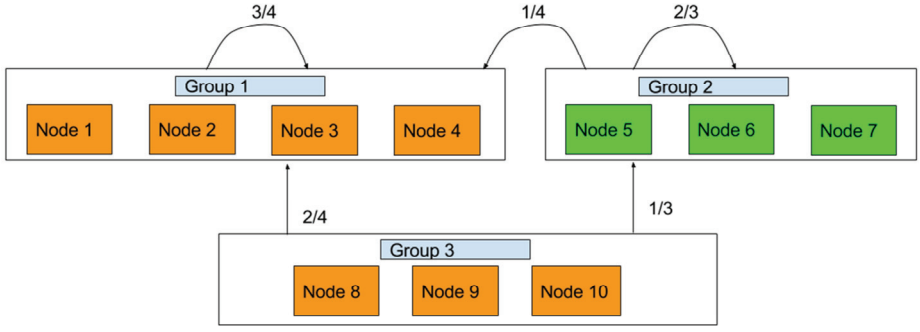

Fig. 10. Example of quorum slice in Stellar.

3.2.2 Crash fault tolerance based consensus

Motivated by the classical and difficult-to-understand consensus algorithm called Paxos [65], Raft [80] is a consensus algorithm that is used by Quorum [81] to tolerate crashes. Being different from Paxos, Raft was created to be easier to understand and use in practical systems. Like Paxos, Raft is made based on an assumption that every time, $[n/2 + 1]$ of the total nodes work normally. To execute the Raft consensus algorithm, verifying nodes can belong to three different roles, which are follower, candidate, and leader. These nodes communicate with each other by two main kinds of message: RequestVote for voting a leader node, and AppendEntries for transferring the requests to other nodes. Fig. 11 describes how nodes change their roles with different situations. Basically, there would be some sequential-

numbered terms of time, in which a node holding leader role will retain its responsibility until the end of the term, then switch back to the follower role. Initially, all the nodes start as follower role. These nodes remain in this state, until they do not receive any AppendEntries messages from the leader node (there is no leader node at first, or the leader is crashed in the following terms). Then after election timeout period, all follower nodes will start election to vote for a leader node, by transferring their role to candidate, and increasing the term number. Each candidate elects their wanted leader by broadcasting a RequestVote message, accompanied by their current term number. When a candidate receives the RequestVote message, first of all, he would have to check if the received term number is equal to his current term number; and if his is larger, the received message is discarded. After aggregating all the messages, if a node receives the vote from the majority of the candidates, he will become leader; otherwise, he will return to the follower role. In the special case that a node could not decide who he will be, because he is voted by 50% of all the nodes, the new election round will start again. The leader will be recognized by every node, because the RequestVote message is broadcasted.

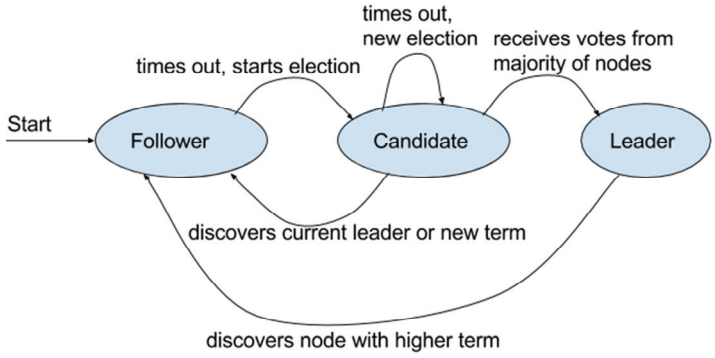

Fig. 11. Node's role transition in Raft [80].

During the consensus execution, the leader receives some requested transactions sent from many clients, then he will write them to a list called log entry. Afterwards, he sends to all followers the AppendEntries message containing each logged transaction *r* and the index of the previous transaction *pi* in the list. For example, if the leader sends the 5th transaction, then he will also have to send the 4^{th} as the index of the previous transaction in the list. When a follower receives AppendEntries message, if *pi* is the index of the last transaction he received, he will write *r* to his log entry list. Otherwise, the leader will have to find the latest transaction that he and the contradicting follower have in common, then this follower will delete all transactions after the found transaction, and start synchronizing the log entry list again with the leader. These processes are made to make sure transactions are ordered the same in all verifying nodes. After knowing that the transaction lists are the same in all nodes, the leader node will choose an index in the list, then commit all the transactions before this index, which verifies the transaction, and put the valid ones into a block. Afterwards, he will append the block into his chain, and broadcast the results to other follower nodes in the network, who will see the result, and make the same commitment like the leader.

Another Blockchain platform called Chain [82], employs the crash fault tolerance consensus algorithm named federated. In this one, among *n* nodes in the verifying network, there is a node called 'block generator', and other nodes called 'block signers'. The block generator receives transactions from

clients, then verifies each of them, and stores the valid ones in a temporary list. After each duration, the block generator will sequentially take some requested transactions to put into blocks, which will be sent to all block signers. These signers will validate the received blocks, and sign into them if they are valid, then send them back to the block generator. If a block is signed by more than m ($m < n$) different block signers, the block generator will append the block to his current chain, and propose this block to other nodes. Following this method, Chain could resist the crash fault if the block signer does not work. However, if this case happens with the *block generator*, the network would be halted.

4. Discussion

Table 2 shows some overall comparisons between the proof-based and vote-based consensus algorithms. First of all, it is evident that executing the vote based consensus algorithm should not be free, which limits the number of nodes in the verifying network. This is because if there are too many nodes joining the verifying network, it would be very complicated to exchange messages between them. Also, if nodes are able to join freely, it would be impossible to make the trust assumption, which makes the ledger unreliable. Meanwhile, with the proof-based consensus algorithm, nodes can join and withdraw from the verifying network whenever they want, which makes the number of nodes executing the consensus algorithm unknown. Therefore, showing enough eligibility to append a new block to the chain is suitable for this freedom to join.

Table 2. Comparisons between vote-based consensus and proof-based consensus algorithms

Criterion	Vote-based consensus algorithm	Proof-based Consensus algorithm
Agreement making basement	From majority of the node decisions	Following nodes performing enough proof (PoW, PoS, etc.)
Nodes can join freely	No	Mostly
Number of nodes executing	Limited	Mostly unlimited
Decentralization	Low	Mostly high
Trust	Less trustful	More trustful
Nodes identities are managed	Yes	No
Security threat	Less serious	More serious
Award	Mostly no	Yes

Moreover, vote-based consensus is often conducted in private and consortium Blockchain, in which the decentralization degree is lower than in public Blockchain with proof-based consensus. Consequently, trust in the proof-based one would be higher. Clearly, the more freely that nodes can join the verifying network, the more decentralized the verifying network is.

In contrast, a tradeoff is recorded between the freedom and security issues. The more nodes that join the verification network, the more risk the verification network has. In vote-based consensus, all the nodes are manageable, which makes it safer to control the security risk. While with the proof-based consensus algorithm, some threats, like double spending attack, could always possibly happen.

Finally, with proof-based consensus algorithm, in order to encourage nodes to maintain the ledger,

award is often given to any nodes who mine a new block. Following Satoshi, the more blocks that are in the chain, the fewer awards a node can receive after mining a new block in the future. We suppose that lately, if the mining award and the incentive for verifying transactions are too small, nodes will have no motivation like current time to maintain the ledger anymore. Meanwhile, with vote-based consensus, there is no necessity to give any award to a node. Using decentralization to increase the reliability is the only aim for this kind of algorithm.

To wrap up all the consensus algorithms surveyed in this paper, Table 3 is presented with brief overview of each algorithm.

Table 3. Summary of surveyed consensus algorithms in this paper

	Brief working mechanism description
Proof-based consensus algorithms	
PoW-based consensus algorithms	
Pure PoW [2]	Nodes have to solve a difficult puzzle to get the right to append new blocks to the chain and get the rewards.
Cuckoo hash function-based PoW [27]	The puzzle in pure PoW is replaced by another task: finding a graph established by a Cuckoo hash function.
Prime number finding-based PoW [28]	The puzzle is finding a chain of Cunningham prime numbers.
Double puzzles-based PoW [35]	Two consequential puzzles are created to prevent pool mining.
Non-outsourceable puzzles [37]	The puzzle is re-designed so miners inside pool can get all the rewards without mining (prevents pool mining).
Bitcoin-NG [38]	When a node creates a key block, he could continuously make some micro blocks with transactions, until another node creates a new key block.
GHOST strategy [39,40]	The forking problem is solved by choosing the brand having most PoW contributes to.
Generalized PoW [41]	Each node will find as many nonces as possible. Then the block adder will be chosen based on a lucky factor with their created nonces.
PoS-based consensus algorithms	
Pure PoS (Nextcoin [8])	After every 60 seconds, a random number is chosen. The node who owns the coin whose index is equal to the chosen number gets the right to append his proposed block.
State of the block-based PoS	Based on the state of the current chain, a random Satoshi (smallest currency unit) is chosen. The miner owning this Satoshi is chosen to append his proposed block to the chain.
PoS by coin flipping from many nodes	Many nodes perform coin-flipping protocol to choose the block adder.
Delegated PoS	A delegation including nodes will be voted by other nodes. The more stake a node owns, the more powerful voting power he has to assign the witness. This group will do the main appending job.
Hybrid forms of PoW and PoS	
Coin age-based PoW difficulty re-designation (PPcoin [49])	A so-called coin age is calculated by multiplying the number of coins a miner has, and the duration he keeps them. Based on the coin age spent on a self-transaction, the node solves a PoW puzzle with different difficulty.
Stake-based PoW difficulty re-designation (Blackcoin [50])	The node solves a PoW puzzle based on the coin he owns.

Table 3. Continued

	Brief working mechanism description
Coin age with an exponential decay function [51]	Coin age is used again in this algorithm, but the increasing speed is lower by the decay function of time.
Combining PoW and PoS to append blocks sequentially [52]	After the PoW puzzle is solved, the solver appends a new so-called PoW block to the chain, and provides a base to choose another stake holder to append another so-called PoS block.
[52] with difficulty adjustment [53]	The PoW and PoS difficulty from [52] are adjusted with the environment.
Proof of activity [54]	Initially, a block without transactions is made with a suitable nonce. Then, N nodes will be chosen randomly by the stake they have. The last one will add transactions to the block.
Other kinds of proof-based consensus algorithms	
Puzzles designed for human PoW [55]	The puzzle is designed in a way that is easy for computers to solve, but difficult for human to do.
Proof of burn [56]	Nodes have to send their coins to an address to mine a new block. The miner sending the largest number of coins get the chance to mine a new block.
Proof of space [14,57]	The algorithm generates some so-called plots. The miner storing the most number of plots can mine a new block.
Proof of elapsed time [12]	All nodes receive different waiting time duration, and the node with shortest duration will mine a new block.
Proof of luck [13]	All the nodes make their own blocks with different lucky numbers, and append it to their chains. The chain having the largest total value of lucky numbers is chosen as the main one.
Multichain [62,63]	Round-robin is used to choose the block appender.
Vote-based consensus algorithms	
Byzantine fault tolerance-based consensus	
Hyperledger with practical Byzantine fault tolerance [67]	This consensus algorithm employs three rounds of PBFT to double-check that the blocks they will append are the same.
Symbiont, R3 Corda with BFT-SMaRt [70,71]	This is similar to PBFT, with the support of restoring after any nodes crash.
Iroha with Sumeragi [73,74]	This algorithm uses chain architecture for nodes to exchange messages.
Ripple [76-78]	Transactions verified by a node are broadcasted to others in his own defined list. Transactions verified by at least 80% of all the nodes are appended to the ledger.
Stellar [79]	If any transactions are verified to be valid by all nodes in a node's quorum slice, this node will confirm these transactions to be valid.
Crash fault tolerance-based consensus	
Quorum with Raft [65]	A synchronization between the leader and other nodes is made to make sure the orders of transactions are the same for all nodes, before executing them.
Chain [82]	A block generator will verify a block, and broadcast it to some other nodes. If this block is verified to be valid by more than a threshold of nodes, it would be appended to the chain.

5. Conclusion

Overall, our survey has highlighted some consensus algorithms used in Blockchain, which are categorized into two main kinds: proof-based and vote-based consensus algorithms. In the former, nodes have to show they have performed sufficient proof to get the right to do the appending work, and get the rewards. Meanwhile, in the latter, nodes will exchange messages with others to make an agreement about the blocks or transactions to be appended to the ledger. We also make comparisons between these two types based on some of their highlighted characteristics, which illustrates the advantages and drawbacks of each category. It could be observed that besides the original public Blockchain with proof-based consensus algorithms, the newly developed consortium and private Blockchain has much potential with vote-based ones at this time.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2017R1A2B4012559).

References

[1] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology,* vol. 3, no. 2, pp. 99-111, 1991.

[2] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008 [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[3] The Economist, "The great chain of being sure about things," 2015 [Online]. Available: https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable.

[4] Bitcoinwiki, "Genesis block," 2017 [Online]. Available: https://en.bitcoin.it/wiki/Genesis_block.

[5] E. Robert, "Digital signatures," 2017 [Online]. Available: http://cs.stanford.edu/people/eroberts/courses/soco/projects/public-key-cryptography/dig_sig.html.

[6] Ethereum [Online]. Available: https://www.ethereum.org.

[7] S. Popov, "A probabilistic analysis of the Nxt forging algorithm," *Ledger*, vol. 1, pp. 69-83, 2016.

[8] Nxt wiki, "Whitepaper: Nxt," 2016 [Online]. Available: https://nxtwiki.org/wiki/Whitepaper:Nxt.

[9] The Public Disputes Program, "A short guide to consensus building," [Online]. Available: http://web.mit.edu/publicdisputes/practice/cbh_ch1.html.

[10] A. Back, "Hashcash: a denial of service counter-measure," 2002 [Online]. Available: ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf.

[11] Bitcoinwiki, "Proof of Stake," 2014 [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_Stake.

[12] Intel, "Sawtooth v1.0.1," 2017 [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html.

[13] M. Milutinovic, W. He, H. Wu, and M. Kanwa, "Proof of luck: an efficient Blockchain consensus protocol," in *Proceedings of the 1st Workshop on System Software for Trusted Execution*, New York, NY, 2016.

[14] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gazi, "Spacecoin: a cryptocurrency based on proofs of space," 2017 [Online]. Available: https://eprint.iacr.org/2015/528

[15] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proceedings of ACM Workshop on Distributed Cryptocurrencies and Consensus Ledgers,* Chicago, IL, 2016.

[16] QuorumChain Consensus [Online]. Available: https://github.com/jpmorganchase/quorum/wiki/Quorum Chain-Consensus.

[17] L. Lamport, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 51-58, 2001.

[18] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, New Orleans, LA, 1999, pp. 173-186.

[19] M. Castro and B. Liskov, "Byzantine fault tolerance," U.S. Patent 6671821 B1, Dec 30, 2003.

[20] C. Pommier, "How the private and public key pair works," 2017 [Online]. Available: https://www.symantec.com/connect/blogs/how-private-and-public-key-pair-works.

[21] Elliptic curve digital signature algorithm, 2017 [Online]. Available: https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm.

[22] Bitcoin Stack Exchange, "Can someone explain how the Bitcoin Blockchain works?," 2017 [Online]. Available: https://bitcoin.stackexchange.com/questions/12427/can-someone-explain-how-the-bitcoin-blockchain-works.

[23] Bitcoinwiki, "Block hashing algorithm," 2015 [Online]. Available: https://en.bitcoin.it/wiki/Block_hashing_algorithm.

[24] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology–CRYPTO'87*. Heidelberg: Springer, 1987, pp. 369-378.

[25] Bitcoinwiki, "SHA-256," 2016 [Online]. Available: https://en.bitcoin.it/wiki/SHA-256.

[26] J. Tromp, "Cuckoo cycle: a memory-hard proof-of-work system," 2015 [Online]. Available: https://eprint.iacr.org/2014/059.pdf.

[27] K. Schwarz, "Cuckoo Hashing," [Online]. Available: http://web.stanford.edu/class/cs166/lectures/13/Small13.pdf.

[28] S. King, "Primecoin: cryptocurrency with prime number proof-of-work," 2013 [Online]. Available: http://primecoin.io/bin/primecoin-paper.pdf.

[29] C. K. Caldwell, "Cunningham chain," 2017 [Online]. Available: http://primes.utm.edu/glossary/xpage/CunninghamChain.html.

[30] M. Liskov, "Fermat primality test," in *Encyclopedia of Cryptography and Security*. Boston, MA: Springer, 2005, pp. 221-221.

[31] Bitcoinwiki, "Irreversible transactions," 2017 [Online]. Available: https://en.bitcoin.it/wiki/Irreversible_Transactions.

[32] Bitcoinwiki, "Weaknesses," 2017 [Online]. Available: https://en.bitcoin.it/wiki/Weaknesses#Attacker_has_a_lot_of_computing_power.

[33] T. Sameeh, "Two new models for double spending attacks on Bitcoin's Blockchain," 2016 [Online]. Available: https://www.deepdotweb.com/2016/12/31/two-new-models-double-spending-attacks-bitcoins-blockchain/.

[34] CoinDesk Inc., "What are bitcoin mining pools?," 2014 [Online]. Available: https://www.coindesk.com/information/get-started-mining-pools/.

[35] I. Eyal and E. G. Sirer, "How to disincentivize large bitcoin mining pools," 2014 [Online]. Available: http://hackingdistributed. com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools.

[36] M. Bastiaan, "Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin," 2015 [Online]. Available: http://referaat.cs.utwente.nl/conference/22/paper/7473/preventingthe-51-attack%20-a-stochastic-analysis-of-two-phase-proof-of-work-in-bitcoin.pdf.

[37] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, 2015, pp. 680-691.

[38] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: a scalable blockchain protocol," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation,* Berkeley, CA, 2016, pp. 45-59.

[39] Y. Sompolinsky and A. Zohar, "Accelerating Bitcoin's transaction processing: fast money grows on trees, not chains," 2013 [Online]. Available: https://eprint.iacr.org/2013/881.pdf.

[40] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*. Heidelberg: Springer, 2015, pp. 507-527.

[41] Z. Liu, S. Tang, S. S. M. Chow, Z. Liu, and Y. Long, "Forking-free hybrid consensus with generalized proof-of-activity," 2017 [Online]. Available: https://eprint.iacr.org/2017/367.pdf.

[42] Bitcoin forum, "Topic: Proof of stake instead of proof of work," 2011 [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0.

[43] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Financial Cryptography and Data Security*. Heidelberg: Springer, 2016, pp. 142-157.

[44] A. Russell and D. Zuckerman, "Perfect information leader election in log* n+O(1) rounds," in *Proceedings 39th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, 1998, pp. 576-583.

[45] M. Ben-Or and N. Linial, "Collective coin flipping," Institute of Mathematics and Computer Science, The Hebrew University, Jerusalem, Israel, 1990.

[46] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: a provably secure proof-of-stake blockchain protocol," 2016 [Online]. Available: https://eprint.iacr.org/2016/889.pdf.

[47] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: a provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology–CRYPTO 2017*. Cham, Switzerland: Springer, 2017, pp. 357-388.

[48] D. Larimer, "Delegated Proof-of-Stake (DPOS)," 2014 [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/.

[49] S. King and S. Nadal, "PPcoin: peer-to-peer crypto-currency with proof-of-stake," 2012 [Online]. Available: https://decred.org/research/king2012.pdf.

[50] P. Vasin, "Blackcoin's proof-of-stake protocol v2," 2014 [Online]. Available: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

[51] L. Ren, "Proof of stake velocity: building the social currency of the digital age," 2014 [Online]. Available: https://www.reddcoin.com/papers/PoSV.pdf.

[52] T. Duong, L. Fan, and H. S. Zhou, "2-hop Blockchain: combining proof-of-work and proof-of-stake securely," 2016 [Online]. Available: https://eprint.iacr.org/2016/716.pdf.

[53] A. Chepurnoy, T. Duong, L. Fan, and H. S. Zhou, "TwinsCoin: a cryptocurrency via proof-of-work and proof-of-stake," 2017 [Online]. Available: https://eprint.iacr.org/2017/232.pdf.

[54] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: extending bitcoin's proof of work via proof of stake," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34-37, 2014.

[55] J. Blocki and H. S. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *Theory of Cryptography*. Heidelberg: Springer, 2016, pp. 517-546.

[56] P4Titan, "Slimcoin: a peer-to-peer crypto-currency with proof-of-burn," 2014 [Online]. Available: http://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf.

[57] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space" in *Advances in Cryptology–CRYPTO 2015*. Heidelberg: Springer, 2015, pp. 585-605.

[58] R. Echevarria, "The second coming of blockchain," 2017 [Online]. Available: https://software.intel.com/en-us/blogs/2017/02/14/the-second-coming-of-blockchain
[59] Hyperledger Sawtooth [Online]. Available: https://sawtooth.hyperledger.org/docs/.
[60] M. Sabt, M. Achemlal and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *Proceedings of 2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, 2015, pp. 57-64.
[61] Intel Software Guard Extensions (Intel SGX) [Online]. Available: https://software.intel.com/en-us/sgx.
[62] Multichain [Online]. Available: https://github.com/MultiChain/multichain.
[63] G. Greenspan, "MultiChain Private Blockchain," 2015 [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf
[64] W. L. Heimerdinger and C. B. Weinstock, "A conceptual framework for system fault tolerance," Defense Technical Information Center, *Technical Report CMU/SEI-92-TR-033*, 1992.
[65] L. Lamport, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 18-25, 2014.
[66] L. Lamport, R. Shostak, and M, Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, 1982.
[67] Hyperledger [Online]. Available: http://hyperledger.org/.
[68] Hyperledger fabric [Online]. Available: https://github.com/hyperledger/fabric.
[69] H. Sukhwani, J. M. Martinez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric)," in *Proceedings of the IEEE 36th Symposium on Reliable Distributed Systems*, Hong Kong, China, 2017, pp. 253-255.
[70] Symbiont [Online]. Available: https://symbiont.io/.
[71] Corda [Online]. Available: https://www.corda.net/.
[72] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMART," in *Proceedings of 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, 2014, pp. 355-362.
[73] Hyperledger iroha [Online]. Available: https://github.com/hyperledger/iroha.
[74] Sumeragi [Online]. Available: https://github.com/hyperledger/iroha/wiki/Sumeragi.
[75] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: byzantine replication with high throughput and embedded reconfiguration," in *Principles of Distributed Systems*. Cham, Switzerland: Springer, 2014, pp. 91-106.
[76] D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," 2014 [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf.
[77] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: overview and outlook," in *Trust and Trustworthy Computing*. Cham, Switzerland: Springer, 2015, pp. 163-180.
[78] Ripple [Online]. Available: from https://ripple.com/.
[79] D. Mazieres, "The Stellar Consensus Protocol: a federated model for internet-level consensus," 2015 [Online]. Available: https://www.stellar.org/papers/stellar-consensus-protocol.pdf.
[80] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of 2014 USENIX Annual Technical Conference*, Philadelphia, PA, 2014, pp. 305-319.
[81] Raft-based consensus for Ethereum/Quorum [Online]. Available: https://github.com/jpmorganchase/quorum/blob/master/raft/doc.md.
[82] Federated Consensus [Online]. Available: https://chain.com/docs/1.2/protocol/papers/federated-consensus.

Giang-Truong Nguyen https://orcid.org/0000-0002-6034-1013

He received B.S. degree in School of Information and Communication Technology from Hanoi University of Science and Technology in 2015. Since March 2017, he is with the School of Electronics and Computer Engineering from Chonnam National University as an M.S. student.

Kyungbaek Kim https://orcid.org/0000-0001-9985-3051

He received B.S., M.S., and Ph.D. degrees in School of Electrical Engineering and Computer Science from Korea Advanced Institute of Science and Technology in 1999, 2001, and 2007, respectively. Since 2012, he is with the School of Electronics and Computer Engineering from Chonnam National University as a Professor.