applied sciences

MDPI

Article

A Compute and Wait in PoW (CW-PoW) Consensus Algorithm for Preserving Energy Consumption

Mostefa Kara [1,*], Abdelkader Laouid [1], Muath AlShaikh [2], Mohammad Hammoudeh [3], Ahcene Bounceur [4,*], Reinhardt Euler [4], Abdelfattah Amamra [5] and Brahim Laouid [1]

1 LIAP Laboratory, El Oued University, P.O. Box 789, El Oued 39000, Algeria; abdelkader-laouid@univ-eloued.dz (A.L.); laouid.bahi@gmail.com (B.L.)
2 Computer Science Department, College of Computing and Informatics, Saudi Electronic University, Riyadh 11673, Saudi Arabia; M.ALSHAIKH@seu.edu.sa
3 School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester M1 5GD, UK; M.Hammoudeh@mmu.ac.uk
4 Lab-STICC UMR CNRS, University of Western Brittany UBO, 6285 Brest, France; Reinhardt.Euler@univ-brest.fr
5 Computer Science Department, California State Polytechnic University Pomona, 3801 W Temple Ave, Pomona, CA 91768, USA; aamamra@cpp.edu
* Correspondence: karamostefa@univ-eloued.dz (M.K.); Ahcene.Bounceur@univ-brest.fr (A.B.)

Abstract: Several trusted tasks use consensus algorithms to solve agreement challenges. Usually, consensus agreements are used to ensure data integrity and reliability in untrusted environments. In many distributed networking fields, the Proof of Work (PoW) consensus algorithm is commonly used. However, the standard PoW mechanism has two main limitations, where the first is the high power consumption and the second is the 51% attack vulnerability. In this paper, we look to improve the PoW consensus protocol by introducing several proof rounds. Any given consensus node should resolve the game of the current round $Round_i$ before participating in the next round $Round_{i+1}$. Any node that resolves the game of $Round_i$ can only pass to the next round if a predetermined number of solutions has been found by other nodes. The obtained evaluation results of this technique show significant improvements in terms of energy consumption and robustness against the 51% and Sybil attacks. By fixing the number of processes, we obtained an energy gain rate of 15.63% with five rounds and a gain rate of 19.91% with ten rounds.

Keywords: improved PoW consensus algorithms; blockchain; energy consumption; distributed systems; game competition

Citation: Kara, M.; Laouid, A.; AlShaikh, M.; Hammoudeh, M.; Bounceur, A.; Euler, R.; Amamra, A.; Laouid, B. A Compute and Wait in PoW (CW-PoW) Consensus Algorithm for Preserving Energy Consumption. *Appl. Sci.* **2021**, *11*, 6750. https://doi.org/10.3390/app11156750

Academic Editor: Gianluca Lax

Received: 30 May 2021
Accepted: 25 June 2021
Published: 22 July 2021

1. Introduction

Distributed computing is a computing field that studies distributed systems (DS) whose components are located on different networked computers spread over different geographies and which communicate by transmitting messages in order to achieve a common goal [1]. In this scenario and with the absence of a global clock, the event of failure of an independent component in the system must tolerate the failure of individual computers. Each computer has only a limited and incomplete view of the system. Various hardware and software architectures are used for DS. In peer-to-peer (P2P), there are no special machines that provide services or manage network resources and the responsibilities are evenly distributed among all machines called peers. Peers can serve as both clients and servers. In the context of DS, the Byzantine Generals Problem (BGP) is a distributed computational problem that was formalized by Leslie Lamport, Robert Shostak and Marshall Pease in 1982 [2]. The BGP is a metaphor that deals with the questioning of the reliability of transmissions and the integrity of the interlocutors. A set of elements working together must indeed manage possible failures between them. These failures will then consist of the presentation of erroneous or inconsistent information. The management of these faulty

components is also called fault tolerance which leads us to talk about synchronous and asynchronous systems. Fischer, Lynch and Paterson (FLP) [3] have shown that consensus can not be reached in a synchronous environment if even a third of the processors are maliciously defective. In an asynchronous system, even with a single faulty process, it is not possible to guarantee that consensus will be reached (the system does not always end). FLP says that consensus will not always be reached, but not that it ever will be. This study concerns asynchronous systems, where all processors operate at unpredictable speeds [4]. Theoretically, a consensus can not always be achieved systematically asynchronously. But despite this result, it is possible to obtain satisfactory results in practice, as for instance by the non-perfect algorithms of Paxos Lamport [5] (in the context of obvious failures and no Byzantine faults). Lamport, Shostak and Pease showed in [2], via the Byzantine Generals Problem, that If f is the number of faulty processes, it takes at least $3f + 1$ processes (in total) for the consensus to be obtained. Under these conditions, the PoW technique has ensured the consensus perfectly, but its major problem is the enormous consumption of energy. In this paper, we propose a consensus protocol for an asynchronous environment. We minimize the energy consumption to ensure the synchronization by the application of several rounds of consensus, where in each round the nodes make a Proof-Wait, i.e., show the PoW then make a wait. The remainder of this paper is organised as follows: In Section 2, we will discuss the consensus problem. In Section 3, we provide an overview of the consensus protocol. In Section 4, we present our proposed protocol. Section 5 demonstrates the validity of our protocol. Section 6 shows the compute and wait implementation. In Section 7, we will discuss the hardness of the proposed cryptosystem. Finally, we conclude with Section 8.

2. Consensus Problem

A fundamental goal in distributed computing and multi-agent systems is to achieve overall system reliability in the presence of a number of faulty processes. This obviously requires the coordination of the correct processes in order to reach an agreement on a final decision. The processes must agree on a common value, where each process must provide a local value which is broadcast to all the other processes (or else, it shows a measure or a calculation). From all the proposed values, the processes must decide on a single common value such that either a leader process initiates the accord phase, or the accord phase is started at predetermined times. Many applications require consensus including blockchain, clock synchronization, cloud computing, opinion-forming, page-rank, smart grids, drone control, state estimation, load balancing and so on.

2.1. Conditions to be validated

The choice of the consensus algorithm is the main element. It determines the level of security and impact, and thus the following points must be achieved:

- Accord: The decided value is the same for all the correct processes.
- Integrity: A process decides at most once and there is no change in value choice.
- Validity: The value chosen by a process is one of the values proposed by the other processes.
- Termination: The decision phase takes place in a finite time (any correct process decides in a finite time).

2.2. Potential attacks

Choosing the wrong consensus algorithm can render the underlying system unusable and put all stored data at risk. The vulnerability of a consensus can expose the system to the following attacks:

- 51% attack: In the PoW algorithm, the domination can be achieved by controlling more than half of the total computation of the network (hash rate) [6]. The pool (a group of miners working together called a mining pool) would be able to add its own blocks to the blockchain or create a competing independent branch to which the main and legitimate branch will converge later. This type of attack notably allows the attacking

pool to be able to spend twice its own funds (double-spending attack) and reject transactions that it does not want to be included in the ledger.
- Sybil attack: One hostile node can conduct a Sybil attack by creating a large number of identities and using them to exert disproportionate influence and to defraud the system to break its trust and redundancy mechanism [7].
- DDoS attack: This attack aims to disrupt the normal functioning of the network by flooding the nodes with information or to lower the expected success outlook of a competing mining pool [8].

3. Consensus Protocol Overview

Bitcoin technology could refer to the most famous blockchain implementation that is created in 2008 by a person or group working under the pseudonym "Satoshi Nakamoto" [9]. In public cryptocurrencies and distributed ledger systems, the fundamental infrastructure of the blockchain is a peer-to-peer overlay network over the Internet [10]. Transactions represent the exchanges between users, and the recorded transactions are grouped together in blocks of size 1M at most. After recording recent transactions, a new block is generated and all transactions will be validated by miners, who will analyze the entire history of the blockchain. If the block is valid, it is time-stamped and integrated into the blockchain. The transactions' contents are then visible on the entire network. Once added to the chain, a block can not be changed or deleted, which guarantees the authenticity and security of the network. Each block in the chain is made up of the following elements: a collection of transactions, the hash (sum of transactions) used as an identifier, the hash of the previous block (except for the first block in the chain, called the genesis block) and the target (a measure of the amount of work that was required to produce the block). The main application of this technology is that of crypto-currencies such as Bitcoin [11]. Beyond its monetary aspect, this decentralized information storage technology could have multiple applications requiring secure exchanges without going through a centralizing body, or unfalsifiable traceability, such as applications based on smart contracts, applications allowing the exchange of all kinds of goods or services, means of improving their predictive systems known as oracles, the traceability of products in the food chain, etc. Each node of the network operates autonomously with respect to the set of rules to which it belongs, and this mechanism of identity management plays a main role in determining the organization of the nodes of a blockchain network.

From the system design perspective, a blockchain network contains four levels of implementation. These are the data and network organization protocols, the distributed consensus protocols, the autonomous organization framework based on intelligent contracts and the application (the interface) [12]. In each type of blockchain, several consensus algorithms are designed. One of the most famous algorithms is Proof of Work (PoW), whose concept was first introduced by Cynthia Dwork and Moni Naor in 1993 [13] and in which the authors have presented a computational technique to combat spam in particular and control access to a shared resource in general. The main idea is to require a user to calculate a moderately difficult but not insoluble function, in order to access the resource, thus avoiding frivolous use. Then the work should be difficult to do for the requester, but easily verifiable for a third party. In 1997, Adam Back implemented the idea with Hashcash, an algorithm to easily produce proofs of work using a hash function (especially SHA-256), and whose main use was electronic mail. The term 'proof of work' has been coined in 1999 by Markus Jakobsson and Ari Juels in their article Proofs of Work and Bread Pudding Protocols [14]. In Bitcoin, to validate a block, the miner had to build a draft of this block (including transactions and payload data), indicate the identifier of the previous block to make the link, and vary a number present in the header called the nonce. By varying this nonce (as well as other parameters in the block), the miner was able to try a gigantic number of possibilities so that the hash of the header produced a suitable result, i.e., a hash starting with a sufficient number of zeroes. Due to the high power consumption of PoW, the Proof of Stake (PoS) is positioned as an alternative. Peercoin was the first cryptocurrency

to use PoS by Sunny King and Scott Nadal in 2012 [15]. PoS asks the user to prove the possession of a certain quantity of cyber money to pretend to validate blocks. To avoid centralization (the richest member would always have an advantage) and the Nothing at Stake attack, many alternatives exist for a move towards more comprehensive consensus mechanisms which use random allocation methods taking into account the age of the coin (as in the case of Peercoin) and depending on the velocity [16] used by the ReddCoin cryptocurrency. The variant that is often considered as one of the most balanced protocols between security, decentralization and network scalability is Delegated Proof of Stake used by the BitShares cryptocurrency [17]. Its selection is based on votes in which the block validator is randomly selected from a group of 101 delegates who have the highest stakes. Proof of Burn (PoB), or Proof of Destruction [18], is an algorithm very similar to PoS. In PoS, the participant sequesters a certain amount of cryptocurrency, which is a necessary collateral to participate in the validation of the network, but if he wishes to leave this network it is possible to recover his initial stake. What PoB and PoS have in common is the fact that block validators must invest their own coins in order to participate in the consensus mechanism. At PoB, this will involve destroying the coins that the participant provided to gain the right to validate network transactions. This system is similar to PoS in that the more coins it burns, the more likely it is to obtain the associated reward. Proof of Burn is offered as an alternative to the classic Proof of Work, but this young technique is criticized by some detractors who consider it a simple waste of tokens. It is the idea of destroying cryptocurrency in order to create it.

There are also many challenges that attempt to replace "Work" in PoW. For example, Proof of eXercise (PoX) in [19], where the challenge is to solve a real eXercise, a scientific computation problem based on a matrix. The authors chose matrix problems because matrices have interesting composability properties that help to solve the difficulty, collaborative verification and pool-mining, and also that matrix problems cover a wide range of useful real-world problems, being a primary abstraction for most scientific computing problems. The miner must solve the following equation:

$$X_1 \circ X_2 \circ ... \circ X_p = Y \quad (1)$$

where X_i and Y are matrices, $\circ$ is an operator, e.g., a product, a sum, etc. Another challenge proposed is Primecoin [20] which, as its name indicates, consists of finding prime numbers instead of finding the nonce.

Proof of Space [21] or Proofs of Capacity (PoC) is a protocol between a prover P and a verifier V which has two distinct phases. After an initialization phase, P is supposed to store data F of size N, while V contains only a few information. At any later time, V can initiate an execution phase of the proof, and at the end, V outputs reject or accept. The authors demanded that V be very efficient in both phases, while P is very efficient in the execution phase as long as it is stored and has random access to the data F. The simplest solution would be for the verifier V to generate a pseudo-random file F of 100 GB and send it to the prover P during an initialization phase. Later, V can ask P to return a few bits of F at random positions, making sure that V stores (at least a large part of) F. Unfortunately, with this solution, V still has to send a huge 100 GB file to P, which makes this approach virtually useless in practice. The PoC scheme which they proposed is based on graphs that are difficult to engrave. During the initialization phase, V sends the description of a hash function to P, which then labels the nodes of a graph that is difficult to engrave using this function. Here, the label of a node is calculated as the hash of the labels of its children. V then calculates a Merkle hash of all the labels and sends this value to P. In the execution phase of the proof, V simply asks P to open the labels corresponding to certain nodes chosen at random.

Proof of Space-Time (PoST) is another consensus algorithm closely related to PoC. PoST [22] differs from proof of capacity in that PoST allows network participants to prove that they have spent a "space-time" resource, meaning that they have allocated storage capacity to the network over a period of time. The authors called this 'Rational' Proofs of

Space-Time because the true cost of storage is proportional to the product of storage capacity and the time that it used. The rational proof of financial interest in the network achieved by PoST addresses two problems with proof of capacity. The first, Arbitrary amortized cost: In a consensus system that doesn't account for time, participants can generate an arbitrary amount of PoC proofs by reusing the same storage space, and lowering their true cost. The second, Misaligned incentives: A rational participant in a PoC system will discard almost all stored data whenever computation costs less than the data storage do. This essentially turns PoC into a partial PoW system, which is potentially more resource-intensive.

An extension of Bitcoin's PoW via PoS is presented in Proof of Activity (PoA) by Bentov et al. [23]. Miners start with PoW and claim their reward. The difference is that the extracted blocks do not contain transactions. They are simply templates with header information and the mining reward address. Once this nearly blank block is mined, the system switches to PoS. The header information is used to select a random group of validators to sign the block. They are coin holders (stakeholders) and the greater the stake held by a validator, the more likely he or she will be selected to sign the new block. Once all the chosen validators have signed the block, it becomes an actual part of the blockchain. If the block remains unsigned by some of the chosen validators for a given time, it is rejected as incomplete and the next winning block is used. Validators are chosen again and this continues until a winning block is signed by all selected validators. The network costs are divided between the winning miner and the validators who signed the block. PoA is criticized that too much power is still needed to mine blocks during the PoW phase on one hand. On the other hand, coin accumulators are even more likely to make the signatory list and rack up more virtual currency rewards.

A random mining group selection to prevent 51% Attacks on Bitcoin is proposed in [24]. The authors divide miners into several groups. Each peer node determines its mining group using the Hg (·) hash function and its wallet address. Additionally, once a block is created, its hash value is used with Hg (·) to determine which mining group is supposed to find the next block. Only even nodes belonging to the mining group are allowed to mine the next block and to compete with each other. Once a block is propagated over the P2P network, other nodes can check if the block was generated by the correct mining group by comparing the hash value of the previous block in the header of the block with the address of block creators. Here, although there may be an attacker with more than half of the total hash power, the chances of a successful double-spend attack can be greatly reduced by increasing the number of mining groups as the mining groups are chosen at random. In addition, the computational power required for block mining is effectively reduced by 1/(number of groups) because even nodes not belonging to the selected group do not participate in PoW and the difficulty level can be lowered due to the smaller number of competing miners in each group. The authors show that if the number of groups is greater than or equal to two, the probability of the attacker of finding the next block is less than 50%.

4. Proposed Protocol

There are three types of blockchain, private, public and hybrid. A private blockchain (permissioned) operates in a restrictive environment, i.e., a closed network. In an authorized blockchain that is under the control of an entity, only authorized nodes with a revealed identity are allowed to enable basic functionalities such as consensus participation or data propagation [25]. Comparatively, in a public blockchain (permission-less/open access), if the node has a valid pseudonym (account address), it can freely join the network and activate any available network functionalities such as sending, receiving and validating transactions and blocks according to common rules. Therefore, there is usually such a blockchain network instance on a global scale that is subject to public governance. Specifically, anyone can participate in the blockchain consensus, although a person's voting power is generally proportional to its possession of network resources, such as computing power, wealth token and storage space [26]. A hybrid blockchain (consortium or federated)

is a creative approach to solving the needs of organizations which have a need for public and private blockchain functionality. Some aspects of organizations are made public, while others remain private. The consensus algorithms in a consortium blockchain are controlled by the predefined nodes. It is not open to the popular masses, it still has a decentralized character and there is not a single centralized force that controls it. Therefore, it offers all the functionalities of a private blockchain, including transparency, confidentiality, and efficiency, without a single party having to consolidate power. In this paper, we are concerned with the second type only, which is the public blockchain.

As shown in Figure 1, we divided the overall operation to reach consensus into several rounds (Supplementary Materials). At the start, a node launches the first round and looks for a solution for its own block, like the basic PoW algorithm, but with a much lower degree of difficulty than what is currently applied. Where the difficulty was X and the number of rounds was 1, in the proposed algorithm, the difficulty is $X' < X$ and the number of rounds is $Y > 1$. Once the solution is found, the node shares it and checks whether there are nine (9) other solutions found in the network for this round (for example, in a scenario of 10 solutions to find). If so, this node has the right to participate in the next round and to restart the PoW. If not, i.e., there are not yet nine (9) solutions found by other nodes, this candidate node will wait until it receives the remaining nine (9) solutions. In the last round, the first node that will find the solution will be the miner, so that in the last round the protocol looks for a single solution. In the original PoW, the proof consists of finding the nonce according to the inequality: $Hash(Block + Nonce) < Target$. In the proposed protocol, the proof of round i is according to the following inequality:

$$Hash(Block + IDRound_{i-1} + Nonce) < Target \quad (2)$$

Initially, the identifier (ID) Round is equal to 1. After that, the *ID Round* is equal to the sum of nonces found in the previous round. Therefore, in each round, there is a new ID so that the nodes will work on it. If we have ten solutions to find in each round, we will obtain the following equation:

$$IDRound_k = \sum_{i=1}^{10} (nonce_i \; of \; round_{k-1}) \quad (3)$$

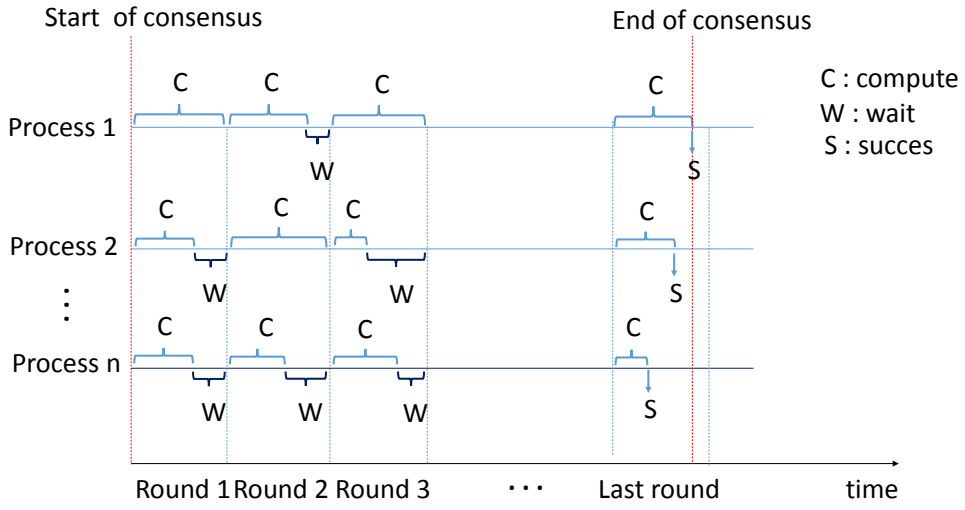

Figure 1. A Compute and Wait Consensus PoW Algorithm.

Let NbrR be the number of rounds and NbrS the number of solutions to find in each round. If two nodes succeed in finding the solution in the last round, then the other nodes will receive two solutions. In this case, the nodes must calculate the standard deviation of

the solutions found in all the rounds for each of these two winners. The miner is the node with the smallest standard deviation. There is another parameter that we can introduce here, which is the consensus state. For the moment, in which round do the processes work? Assuming that a process decides to leave competition if the other processes exceed it by 5 rounds (or 4, for example), this will minimize the energy to be consumed. For a process, if the other competitors overtake him by two rounds, there is little hope of catching them. This process will lose energy unnecessarily if it continues the calculation.

5. Protocol Demonstration

The four conditions of a consensus are: accord, integrity, validity and termination. In fact, we see that there is a fifth condition that must be satisfied in every consensus to be perfect. This condition is equality of opportunity (no domination). In the following, We explain how our protocol verifies these five conditions.

1. Accord: In the final round, in the event that there is only one winner, all the nodes will agree on him. In the case where there is more than one winner, for each winner, we will calculate the standard deviation of his solutions found during all the rounds and we will choose the minimum of these standard deviations. Mathematically, this value is unique. In case there are two branches of the blockchain (fork problem), this problem is solved in basic PoW by the golden rule, where the nodes will choose the longest chain. Finally, the decided value is the same for all the correct nodes and in all cases.
2. Integrity: Once the P_x process obtains a valid nonce $Hash\ (Block_i + Nonce_1) < Target$, the process broadcasts it over the network with a specific date (timestamp). Of course, this nonce is concerned with a specific block (B_{i+1}) and Nonce (N_1), i.e., $P_x(B_{i+1}, N_1)$. After having obtained a second nonce N_2, the process will broadcast it ($P_x(B_{i+1}, N_2)$), and we notice here that there are two nonces N_1 and N_2 for the same block B_{i+1}. In this case, there are several actions we can take, including to choose the smallest nonce. Eventually, each process will participate with at most one value, so we have fulfilled the condition of integrity.
3. Validity: For each solution received, the node will check its validity. The miner at the end is a network node, having a unique public address and resolving the PoW in all rounds.
4. Termination: Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash [6]. PoW resolution is estimated at 10 min, and these 10 min are more than enough for the propagation in the network, and the candidate block will have been added to the blockchain. In the proposed technique, the difficulty is minimized and the number of rounds is increased. To ensure the termination in a finite time, we just need to balance between the two factors difficulty and rounds. So, if the PoW has a termination, the proposed protocol also has a termination.

6. Implementation

In our experiment, we simulated each node by a process, where we implemented multi-process programming. In several tests, we created a different number of processes, each of them starting with the creation of a random number $blockID \leftarrow random()$, which is considered as the ID of the candidate block that the process is working on. After that, each process follows the execution of the proposed algorithm (Algorithm 1). Using five rounds and ten processes, the test took about 5 min, and we got the result shown in Figure 2. The execution at University of El Oued was defined in Python using a mainframe made up of 32 dual-processor nodes of 10 cores each. The intensive computing unit has a management node with the following specifications: Processor: Intel Xeon (R) E5-2660 v3 @ 2.60 GHz x 20, memory: 64 GB of RAM, disk: 2 TB HDD, OS: RedHat Enterprise Linux Server 7.2, OS type: 64 bit. The computing unit has 32 nodes. Each node has the following

characteristics: 10 physical cores, storage capacity: 500 GB HDD and available memory: 64 GB. The Proposed technique can be defined by the following algorithm:

Algorithm 1 Consensus algorithm

Require: $NbrS,\ NbrR$
Ensure: $solution$

procedure CONS($NbrS,\ NbrR$)
  $sols \leftarrow 0$
  $roundID \leftarrow 0$
  $currentRound \leftarrow 1$
  $q \leftarrow Queue()$
  $blockID \leftarrow random()$
  while $currentRound <= NbrR$ **do**
    while $solution\ not\ found$ **do**
      $search\ for\ solution$
    end while
    $sols \leftarrow sols + 1$
    if $currentRound\ is\ NbrR$ **then**
      $print(I\ am\ the\ winner)\ ,\ Exit()$
    end if
    $Broadcast(currentRound,\ solution)$
    $put\ on(q,\ solution)$
    while $sols < NbrS$ **do**
      $Nothing$
    end while
    $sols \leftarrow 0$
    $currentRound \leftarrow currentRound + 1$
    $RoundID \leftarrow \sum_{i=1}^{NbrS} q(i)$
    $Broadcast(currentRound,\ RoundID)$
  end while
end procedure

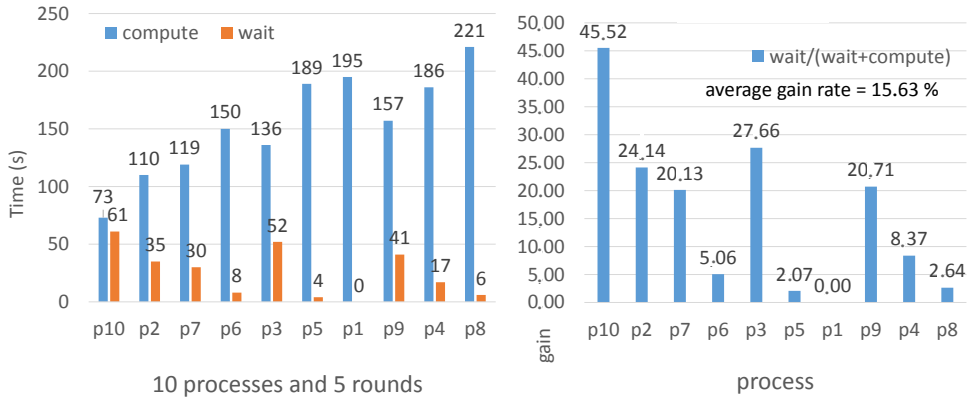

Figure 2. Compute and wait example execution.

We repeated the experiment tens of times in each scenario (each input variation). For example, in Figure 3 with the first scenario where the number of processes is equal to 5 and the number of rounds equal to 5, the gain was 12.07. We repeated this test several times and we obtained different values (between 11 and 14), but we took the most frequent value which was 12.07. We did this at each input change (NbrR, NbrP = (5, 5); (5, 10); (5, 15); (5, 20). We did not take the average of measurements, but we considered the most frequent value.

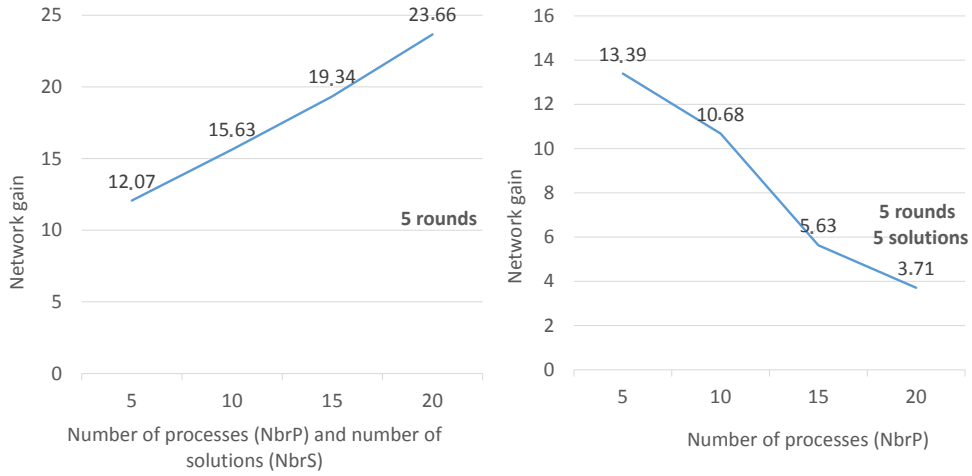

Figure 3. The influence of NbrP for fixed NbrR.

The formal definition of gain: In each round, the process searches for a solution (compute time). Then the process will wait until it receives the latest solution. We will consider this wait time as a gain, and we can formulate the gain rate as follows: *gain rate = wait time/total execution time*. If the total execution time is the wait and compute time, we obtain:

$$gain\ rate = wait\ time/(wait\ time + compute\ time) \tag{4}$$

We let $C_{i,x}$ (respectively, $W_{i,x}$) be the compute (respectively, wait) time of the process i in round x. The gain of the process i can be formulated as follows:

$$gain_i = \sum_{j=1}^{NbrR} (W_{i,j}/(W_{i,j} + C_{i,j})) \tag{5}$$

The average gain rate (network gain rate) is:

$$Gain = \sum_{i=1}^{NbrP} (gain_i/NbrP) \tag{6}$$

In the scenario shown in Figure 2, we have set the number of rounds $NbrR = 5$, $NbrP = NbrS = 10$, where NbrP (respectively, NbrS) is the number of processes (respectively, of solutions). We do not put different values between $NbrP$ and $NbrS$ in order to study the pure effect of the number of rounds. We notice that there is a real compute and wait in each round for most of the processes. Process 10 is the fastest because it has the minimum execution time and the maximum waiting time, that is why it has the highest gain rate (45.52%). On the other hand, process 1 $(p1)$ is the heaviest process, because $p1$ has the highest execution and the lowest waiting time, and thus the lowest gain rate. For $p1$, the gain rate = wait/(wait + execution) = 0/(0 + 195) = 0. We explain that $p1$ is the last process to find the solution, it wakes up the others, and then starts directly the next round without making any wait. Even though $p1$ didn't wait, its computing time is less than the one of $p8$. We explain this by two possibilities. The first is given when $p8$ started, the second possibility when $p1$ quickly found the solution in the last round (there is no wait after the last round) and $p8$ took a long time in the last round. In general, the average gain of the network is 15.63%.

Compute and wait implementation: In the original PoW, each process will continue the calculation for 10 min (until one of the processes finds the hash). In the proposed protocol there are two types of processes, winning processes, which are processes that participate in all rounds (noting that the final winner is the process that has the minimum standard deviation of his solutions). The other type are those processes that leave the

competition after such rounds (for example, when the number of solutions of the next round is equal to NbrS). In these two types, no process does the calculation during 10 min, there is a proof-wait (first type) or a proof-abandon (second type). So, there are two main factors that must be handled, the number of rounds $NbrR$ and the number of solutions to be found in each round $NbrS$.

Lemma 1. *If* $NbrR > 1$ *then* $Gain > 0$

Proof. Based on the randomness of the hash function (Block + Nonce) < Target and on the arbitrary speed of each process, we have:

$$C_{i,x} \neq C_{j,x},\ \forall i,j \in \{1,...,NbrP\},\ \forall x \in \{1,...,NbrR\} \tag{7}$$

where $NbrP$ is the number of processes and $0 < x < NbrR$. According to Equation (7), $|C_{i,x} - C_{j,x}| > 0$, so either $W_{i,x} > 0$ or $W_{j,x} > 0$, which implies that $W_{i,x}/(W_{i,x} + C_{i,x}) > 0$ or $W_{i,x}/(W_{j,x} + C_{j,x}) > 0$. We obtain $Gain > 0$. □

Lemma 2. *If NbrP increases then Gain increases when* $NbrP = NbrS$

Proof. We let the probability of finding the solution by 5 (respectively, 10) processes be P_5 (respectively, P_{10}), we have $P_{10} > P_5$. Let T_p be the compute time to find the solution with a probability p. So, $T_{p_{10}} < T_{p_5}$ implies that the wait time $W_{p_{10}} > W_{p_5}$. According to Equation (4), $gain_{10processes} > gain_{5processes}$. □

Discussion: In the first scenario shown by Figure 3, we fixed NbrR to 5, and we made several tests, during which we increased the number of processes (NbrP = 5, 10, 15, 20) to study the effect of this increase and its influence on the gain. These tests have shown that the more many processes do compute and wait, the more the gain increases. We explain this by the following example: in the round x, if first process p_i finds a solution after 4 min, we suppose that p_i will wait $t_1 = 1$ minute to start the next round $x + 1$, so the gain is $1/(1 + 4) = 20\%$. With $NbrP = 10$, the first process p_j will find a solution in 3 min, p_j will wait for $t2 > t1$ ($t2 = 2$ min), so the gain is 2 / (2 + 4) = 30%.

In the second curve (curve on the right in Figure 3), where the NbrS is constant, we observe that the gain decreases, because the number of processes (which are waiting) decreases compared to the total number of processes. On the other hand, the number of processes that do not wait is increasing, which implies a decrease in the average gain in the network.

Lemma 3. *While* $NbrR <$ *upper bound, if NbrR increases then Gain increases.*

Proof. Let w be the wait time, c the compute time, and e the execution time, so $e = w + c$. The gain $g = \frac{w}{e}$, and if a process does little computation and a lot of wait, then it will have a high gain rate. We know that there is no wait time in the last round, therefore, if $NbrR = 2$, then $g = \frac{w}{e_1+e_2}$ where w is the wait time of round 1, e_1 the compute time of round 1, and e_2 the compute time of round 2. Let $e_1 + e_2 = 2 \times e$. If $NbrR = 3$, then $g = \frac{2\times w}{3\times e}$, if $NbrR = 4$, then $g = \frac{3\times w}{4\times e} \ldots$, if $NbrR = \alpha$, then $g = \frac{\alpha\times w}{(\alpha+1)\times e}$. The effect of the wait time absence in the last round decreases if we increase $NbrR$ and the gain g converges to $\frac{w}{e}$ because $\frac{\alpha}{(\alpha+1)}$ converges to 1. □

Discussion: In the second scenario illustrated by Figure 4, we fixed $NbrP$ to 10, and we made several tests, in each of them increasing the number of rounds ($NbrR = 5, 10, 15, 20$) to study the effect of this increase and its influence on the gain. These tests have shown that the more $NbrR$ increases, the more the gain increases but not absolutely. Rather, this increase converges to an upper bound related to the number of processes. Then the gain decreases because the difficulty is also reduced (NbrR increases then difficulty decreases).

Therefore, the processes will end almost at the same time because the solution is easy to find.

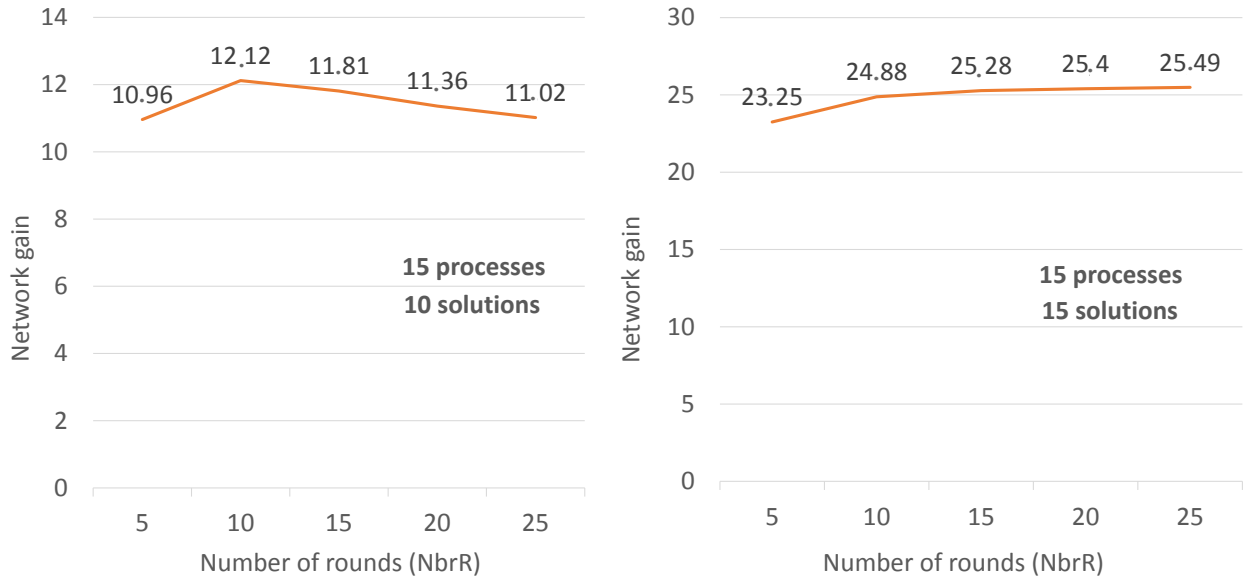

Figure 4. The influence of NbrR, case A.

Lemma 4. *If NbrR increases then* $Gain(G_{nw})$ *converges to a number G*

Proof. The average gain of the network $(G_{nw}) = (g_1 + g_1 + \ldots g_{NbrR})/NbrR$ where g_i is the gain of round i. We have g_{NbrR} (gain of the last round) equal to 0 because there is a calculation and there is no wait. g_i depends on moments $t_{i,j}$ at which solutions are found ($t_{i,1}$ is the instant where the first solution of round i is found, $t_{i,NbrR}$ is the instant at which the last solution of round i is found). In general, these times are random in each round, therefore, $g_1 \approx g_2 \approx \ldots g_{NbrR-1}$ and then $G_{nw} = (NbrR - 1) \times g_1/NbrR$ □

We obtain:

$$G_{nw} = g_1 - (g_1/NbrR) \tag{8}$$

We observe that if $NbrR = 1$ then $G_{nw} = 0$ and if $NbrR$ *increases* then $G_{nw} \approx g_1 = G$

Discussion: Figure 5 shows how the gain converges to a constant number G when the number of rounds increases (NbrR = 5, 10, 15, 20, 25). Usually, G depends on the number of processes and the number of solutions. In Figure 5, we have fixed the number of processes and the number of solutions. In Figure 5-right, (NbrP, NbrS) = (10, 5) and we have set up several scenarios (NbrR = 5, ... until NbrR = 200). We noticed that starting from NbrR = 20 the gain remains greater than 8 and less than 9. In the similar way, in Figure 5-left, the gain remains close to 19.5 whenever NbrR $>$ 20 (20 $\leqq$ NbrR $\leqq$ 200). This is why we said that the gain converges to a constant number G when the number of rounds increases. $G \approx 8.5$ with (NbrP, NbrS) = (10, 5) and NbrR $\geqq$ 20; $G \approx 19.5$ with (NbrP, NbrS) = (10, 10) and NbrR $\geqq$ 20.

The number of solutions is a very important factor because it designates the number of processes that will wait; therefore, it controls the gain. Figure 6 shows the increase in gain as NbrS increases. On the other hand, we cannot introduce a number of solutions without having a multi-round environment. The importance of NbrS directs us towards private blockchains; In which we can talk about NbrS compared to the total number of processes. In another philosophy, we may consider that the node leaves the competition if NbrS of the next round is saturated. In this case, we will increase the gain to the maximum (Table 1). The downside of this philosophy is that after a round j in which the number

of processes is NbrS and one of these processes breaks down, the rest of the competitors (NbrS-1) cannot continue because NbrS (in round $j + 1$) will never be reached.

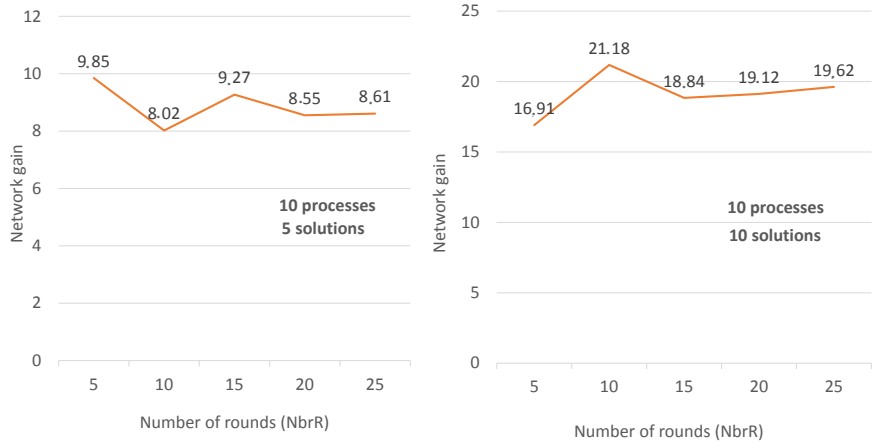

Figure 5. The influence of NbrR, case B.

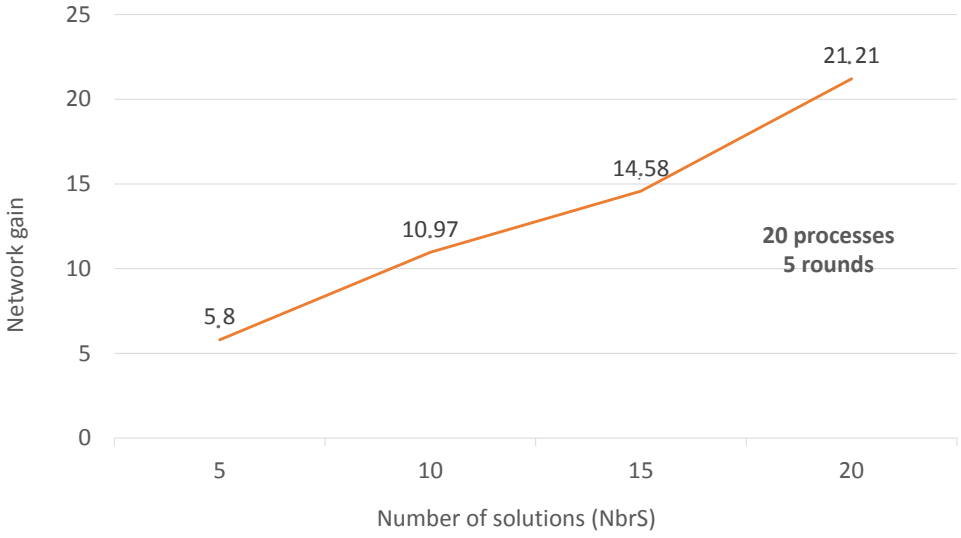

Figure 6. The influence of NbrS.

Table 1. The gain rate with leaving competition.

NbrR	NbrS	NbrP	Gain Rate
5	5	10	55
5	5	20	60
5	5	30	70
5	5	100	90

7. Analysis

To increase the level of security, PoW is based on the degree of difficulty. However, it remains weak against 51% attacks [27]. Regarding domination, when the difficulty is equal to D, there will be only one candidate (N_1) who has the greatest computing power. We have already done a test and reduced the difficulty to 50% to obtain 10 candidates, so the probability that N_1 wins is 10% instead of 100%, because no process controls the standard deviation of its solutions. Therefore, the dominance has been reduced.

As for the 51% attack, when dominance is reduced, the 51% attack will also be reduced. On the other hand, the pool can not work freely to find a single solution, it has to go through several rounds. After each round, the node must share the identifier of this round (id_{round}) which is calculated from the solutions trounced in the previous round, from which the protocol implies to introduce this id_{round} in the next proof of round (Equation (3)). These steps will slow down the operation of producing another longer chain (branch) for use in double-spending.

Speaking of the Sybil attack, it is in principle impracticable unless the attacker has more than 50% of the network's computing resources. Also, the consensus mechanism does not prevent an attacker from disrupting the network by creating a number of malicious nodes (false identities). The proposed protocol combines two techniques, multi-rounds and standard deviation. In comparison to the basic PoW, it is difficult for these two techniques and for a false identity to be the final winner. These false identities must create a chain longer than the chain known in the network. The first technique (multi-rounds) will slow down their work, the second (standard deviation) will decrease their chances.

8. Conclusions and Future Work

In this work, we have proposed an effective and applicable consensus algorithm, and shown that, in blockchain or in any setting where we need an agreement, it is adaptable. We have studied its validity according to the four known conditions of the agreement. We have shown the energy gain achieved by this protocol to reach a drop of 15.63% with five rounds and to reach a drop of 19.91% with ten rounds. We have set up several scenarios establishing in each one several tests, with the manipulation of three factors (number of rounds, number of processes and number of solutions in each round). We have studied separately the influence of each factor on the energy consumed, and also, the influence of introducing these factors in comparison to the basic PoW. We have seen the robustness of the algorithm against the most famous attacks (51% and Sybil attack). In the future, we intend to implement this algorithm in other sectors such as healthcare, for instance.

Supplementary Materials: Compute and Wait Proof o Work (CW-PoW) Video is available online at https://mmutube.mmu.ac.uk/media/POW_Demonstration/1_zrjr8fqq.

Author Contributions: Formal analysis, A.B.; Methodology, M.K., A.L. and R.E.; Software, M.H.; Supervision, A.A.; Validation, M.A. and B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Garcia-Molina, H. Elections in a distributed computing system. *IEEE Trans. Comput.* **1982**, *C-31*, 48–59. [CrossRef]
2. Lamport, L.; Shostak, R.; Pease, M. The Byzantine generals problem. In *Concurrency: The Works of Leslie Lamport*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 203–226.
3. Fischer, M.J.; Lynch, N.A.; Paterson, M.S. Impossibility of distributed consensus with one faulty process. *J. ACM (JACM)* **1985**, *32*, 374–382. [CrossRef]
4. Turek, J.; Shasha, D. The many faces of consensus in distributed systems. *Computer* **1992**, *25*, 8–17. [CrossRef]
5. Lamport, L. The part-time parliament. In *Concurrency: The Works of Leslie Lamport*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 277–317.
6. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2019; pp. 1–9. Available online: www.bitcoin.org (accessed on 20 May 2021).
7. Zhang, S.; Lee, J.H. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Trans. Ind. Inform.* **2019**, *15*, 5715–5722. [CrossRef]
8. Johnson, B.; Laszka, A.; Grossklags, J.; Vasek, M.; Moore, T. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 3–7 March 2014; pp. 72–86.

9. Nakamoto, S.A Peer-to-Peer Electronic Cash System. Bitcoin—URL. 2008; Volume 4. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 20 May 2021).
10. Rathi, V.K.; Chaudhary, V.; Rajput, N.K.; Ahuja, B.; Jaiswal, A.K.; Gupta, D.; Elhoseny, M.; Hammoudeh, M. A Blockchain-Enabled Multi Domain Edge Computing Orchestrator. *IEEE Internet Things Mag.* **2020**, *3*, 30–36. [CrossRef]
11. Mudassir, M.; Bennbaia, S.; Unal, D.; Hammoudeh, M. Time-series forecasting of Bitcoin prices using high-dimensional features: A machine learning approach. *Neural Comput. Appl.* **2020**, 1–15. [CrossRef] [PubMed]
12. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access* **2019**, *7*, 22328–22370. [CrossRef]
13. Dwork, C.; Naor, M. Pricing via processing or combatting junk mail. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 1992; pp. 139–147.
14. Jakobsson, M.; Juels, A. Proofs of work and bread pudding protocols. In *Secure Information Networks*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 258–272.
15. King, S.; Nadal, S. *Ppcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*; Self-Published Paper, August; Peace Consortium of Defense Academies and Security Studies Institutes: Garmisch-Partenkirchen, Germany, 2012; Volume 19, pp. 1–6.
16. Ren, L. Proof of Stake Velocity: Building the Social Currency of the Digital Age. Self-Published White Paper. 2014; pp. 1–13. Available online: www.reddcoin.com (accessed on 20 May 2021).
17. Larimer, D. Delegated Proof-of-Stake (dpos). Bitshare Whitepaper. 2014. Available online: http://107.170.30.182/security/delegated-proof-of-stake.php (accessed on 20 May 2021).
18. Karantias, K.; Kiayias, A.; Zindros, D. Proof-of-burn. In Proceedings of the International Conference on Financial Cryptography and Data Security, Kota Kinabalu, Malaysia, 10–14 February 2020; pp. 523–540.
19. Shoker, A. Sustainable blockchain through proof of exercise. In Proceedings of the IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 30 October–1 November 2017; pp. 1–9.
20. King, S. Primecoin: Cryptocurrency with prime number proof-of-work. *citeseerx.ist.psu.edu* **2013**, *1*, 1–6.
21. Dziembowski, S.; Faust, S.; Kolmogorov, V.; Pietrzak, K. Proofs of space. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2015; pp. 585–605.
22. Moran, T.; Orlov, I. Simple proofs of space-time and rational proofs of storage. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; pp. 381–409.
23. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM Sigmetrics Perform. Eval. Rev.* **2014**, *42*, 34–37. [CrossRef]
24. Bae, J.; Lim, H. Random mining group selection to prevent 51% attacks on bitcoin. In Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Luxembourg, 25–28 June 2018; pp. 81–82.
25. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [CrossRef]
26. Unal, D.; Hammoudeh, M.; Kiraz, M.S. Policy specification and verification for blockchain and smart contracts in 5G networks. *ICT Express.* **2020**, *6*, 43–47. [CrossRef]
27. Shi, N. A new proof-of-work mechanism for bitcoin. *Financ. Innov.* **2016**, *2*, 1–8. [CrossRef]