Towards Possibilities of Energy Minimization in Consensus and Mining Paradigm

Lydia Ouaili, Soumya Banerjee, Elena Kornyshova

Towards Possibilities of Energy Minimization in Consensus and Mining Paradigm

Lydia Ouaili
CNAM, Trasna
Paris, France
lydia.ouaili@lecnam.net

Soumya Banerjee
Trasna
Arklow, Irland
soumya.banerjee@trasna.io

Elena Kornyshova
CNAM
Paris, France
elena.kornyshova@cnam.fr

***Abstract*—Energy consumption of consensus mechanisms used in permissionless distributed ledger, such as Bitcoin and Ethereum, has become a popular research topic. Computation and storage operations combined with proof of work (PoW) to prevent attacks requires a lot of energy consumption. Due to the permissionless nature of blockchain, it is difficult to measure the energy consumed by each node. Therefore, estimation methods provide approximations such as lower and upper bounds on energy consumption. A recent work in this direction has shown that PoW is the most energy consuming consensus. In this context, there is a timely requirement to find solutions to optimize the energy consumption. Our proposal aims at building a typical consensus in distributed ledger with an asynchronous protocol to reduce wasted computations involved in the proof of work. For this purpose, we rely on a similarity search index to prevent double-spending attacks and reduce the difficulty mining involved in the PoW, by exploring how it influences the number of required iterations needed to solve the PoW.**

***Index Terms*—Blockchain, Consensus Algorithm, Mining process, difficulty mining, Double spending attack, Energy consumption, Optimization.**

I. Introduction

The blockchain is a type of distributed ledger technology (DLT) composed of several blocks linked together containing any kind of data. The blockchain was introduced the first time in 2008 in the context of the cryptocurrency Bitcoin [1] to store the digitally signed transactions that are issued by users called nodes through their public/private keys. What differentiates this registry from the others is, its decentralization, persistence, anonymity, and auditability [2]. There is no need to validate transactions by a central authority (e.g., a bank), moreover, transactions are visibles to all the network users, and must be confirmed to be added to the block. It is almost impossible to falsify a block and add it to the blockchain since each block is verified and validated by other nodes. Due to the nature of the permissionless networks, where nodes can participate and leave without any control, the blocks are protected by the consensus of the entire network through the Merkle Tree used to verify the integrity of the block transactions and the Proof of Work (PoW) used during the mining process to prevent fraud as double-spending and allows the immutability of the ledger, which enables trust. The nodes that interact for all these activities form a blockchain network and communicate via a generated address. The blockchain enables the history and origin of the transactions since each validated block is recorded with a timestamp. It is based on a high degree of replication of information and computations, relying on techniques that are the subject of much criticism from academia and the industry because of their energy consumption.

Determining the exact value of energy consumption in permissionless DLT is a difficult task since the exact number of nodes that can participate without any control and the characteristics of their hardware are unknown. A significant focus is done on the consensus mechanism as PoW, in particular the mining process where hashing to prevent attacks is ubiquitous. One of the most cited work in this direction is [3], the authors consider different type of hardware to analyze the energy consumption of the mining process and show that the power demand is comparable to a country electricity consumption as Ireland. In a recent work [5], the authors use a prediction model to measure the expected energy consumption of different consensus mechanisms per transaction and confirms the concerns around the energy consumption of PoW, and their results highlight the urgent need to modernize PoW-based blockchains. In this context, we explore possibilities to minimize brute force search involved in the mining process, the most important task in PoW, while preserving the security concerns such as the prevention of double spending. Our aim, in this work, is to investigate optimization steps to reduce the mining process involved in PoW. This paper makes the following contributions:

- Propose a double spending prevention step in the consensus of a blockchain.
- Propose a collaborative proof of work only for transactions that have not been detected as malicious.
- Explore how to adjust the difficulty mining by an experimental approach to reduce the iterations involved in mining process.

Section II introduces some technical background related to the block generation in the bitcoin consensus to highlight the concerns around the energy consumption of PoW. Hence, we specify the intuition and the purpose behind the use of PoW to explain the considerations of our model. Then, in Section III, we introduce the model formulation and the double-spending prevention steps to adjust the difficulty of mining. Finally, in Section IV we perform an experimental analysis of the adjusting issues and conclude our study in Section V with

potential directions for future research.

II. Background

This section focuses on the technical background of the steps involved in the consensus of the bitcoin blockchain and highlights which steps in PoW that we focus on.

In distributed systems, maintaining and securing a dynamic data registry that must be consistent and adapted to the asynchronous environment without depending on a third party is a very difficult task. It depends on the context in which this register is used. In a digital currency system as Bitcoin, double-spending is considered an attack because it is a fraudulent act in which the same currency is spent more than once. The PoW and the broadcast between the peers maintain the consistency of the blockchain to prevent double spending, because if an attacker wants to modify the transaction, he has to redo the work done previously by the miners.

Bitcoin is a system of electronic payment transfer between peers in a decentralized way. Peers issue transactions by means of digital signatures and pseudonym addresses associated with unique public/private key pairs. Transactions are broadcasted in the network, following a particular format that depends on several parameters: the inputs that refer to an output from a previous transaction and the output which contains information about the transfer of currency, such as the list of addresses that can receive the coins. When the transaction is broadcasted to all the peers, it is checked at the time of the reception [1]. It must be correctly formed and has not been previously spent in a block of the blockchain. If these conditions are fulfilled, then the transaction is stored locally in the mempool of nodes. When a transaction is confirmed, it is removed from the mempool. Each node maintains and manages its own mempool, and this depends on its own storage capacity for unconfirmed transactions. When a transaction is confirmed by a miner and included in a block, it is removed from the mempool [6], [7]. The block contains the following fundamental data [2]: block version which indicates the validation protocol to follow, the hash value of header hash of the previous block, the Merkle tree root hash which refers to the hash value of all the transactions in the block, the current timestamp, nBits which refers to the current hashing target displayed in a compact format, and a particular value called nonce, when it is hashed with the Merkle hash root, that must be lower than a certain given number of 256-bit.

PoW is a consensus strategy, more precisely hashcash PoW [8], where peers particularly miners need to find a value called nonce (a 32-bit number) such that the SHA-256 hash of this nonce concatenated with additional data as the Merkle root hash must be less than or equal a certain value called target [1], [2], which translates into a mathematical inequality presented in the section III. This challenge leads miners to perform brute-force search independently of each other to test different nonces until the inequality is satisfied. When one node succeeds, all other nodes must mutually confirm whether it is actually less than the target or not. The brute-force search wastes too much resources.

Many public blockchain technologies rely on proof of work. This is the case for the many derivatives of Bitcoin as well as its biggest competitor, Ethereum. However, alternatives to proof of work exist [2]. The most common alternative is Proof of Stake (PoS) whose energy consumption is very significantly reduced. In proof of stake, mining cost is nearly zero and the probability of being chosen as a validator of a block depends strongly on the amount of cryptocurrency owned (and/or the duration of this ownership). Therefore, the resistance to attacks is based on the value owned (in cryptocurrency) and not on the mobilized computing capacity. However, this is not a fair system since in this case the selection of validators is based on the richest who dominates the network, which is contrary to the principle of decentralization of bitcoin.

When it comes to energy consumption of DLT, in particular the blockchain, there are several research axis, among them, those that analyze and estimate the energy consumption of consensus mechanisms to identify the steps involved in PoW and PoS consuming the most energy [4], [5], [9]. Another research axis focus on reducing this energy by proposing optimized and efficient consensus algorithms [10], [11], [12]. This paper focus on reducing the energy consumption by adjusting the difficulty mining of PoW, which can be classified in the second category of research axis mentioned above.

In order to optimize the energy consumption of PoW, it is necessary to focus first on the interest of its use (security purpose) and the main step involved in PoW (mining process) which requires a lot of energy resources (total power to mine a block in 2014 is between 0.1GW and 10GW [3]).

The main interest of using PoW is to preserve the security and consistency of the blockchain. There is a kind of trade-off between security and proof of work, if an attacker wants to alter the blockchain to spend bitcoins fraudulently he has to redo all the work done by the miners from the first block to the last. This requires a huge amount of time and resources from the attacker, and during this time, the honest miners will have already made progress in expanding the blockchain. In the mining process involved in PoW, each miner makes random nonce attempts to find a valid hash and this is where the most energy is consumed. Energy estimate depends on several factors [9] such as the performance of the hardware differing from one miner to another (e.g. CPU and GPU) and the parameters involved in the validation of the inequality, such as the difficulty mining which measures how hard it is to find a nonce (in other words how hard it is to generate a block), the hashrate which represents the total number of hashes tested for different nonces in one second. To reduce the energy consumed, a recent work [10] proposes a more ecological PoW called Green-PoW, where the authors propose an algorithm where the mining is done by selection of groups that mine, their selection system reduces almost 50% energy consumption while preserving the security level. In this paper we explore another criterion to optimize the energy consumed in the mining process, which is to adjust the difficulty mining to decrease iterations involved in finding the right nonce, and therefore reduce the hash test to validate the inequality which

consequently will reduce energy consumption.

III. Similarity-Index Approach for the Double-Spending Attacks Prevention

To propose a model that minimizes the number of iterations involved in mining we must ensure that it meets the considerations and reasons for which PoW is used (security goal). In our model we first focus on generate a block with non-conflicting transactions, and then perform a mining process.

A double-spending attack requires 3 scenarios, the goal of the attacker is to get a service without spending a coin, so he sends two transactions $\mathcal{TR}_A$ and $\mathcal{TR}_V$, where the reception time of $\mathcal{TR}_V$ intended for the seller must be less than that of $\mathcal{TR}_A$ (intended for the attacker), if the $\mathcal{TR}_V$ transaction is accepted by the seller and the majority of miners are working on the $\mathcal{TR}_A$ transaction to display it in the blockchain, and the time for the vendor to provide the service does not allow him to detect this behavior, the double-spending attack is successful. To avoid this kind of situations, for which proof of work is used, we first focus on verification and search means for conflicting transactions before doing any mining process. First, we model a conflicting transaction according to the [6] framework and then we define the notion of similarity between transaction pairs.

We follow the notations of [6] to model the scenario of a double attack. Let $\mathcal{A}$ and $\mathcal{V}$ represent respectively an attacker and a vendor, the goal of $\mathcal{A}$ is to get a service from $\mathcal{V}$ without spending its Bitcoin (BTC). In order to do this, $\mathcal{A}$ creates two transactions $\mathcal{TR}_A$ and $\mathcal{TR}_V$ which have the same BTC. Let $t_i^{\mathcal{A}}$ and $t_i^{\mathcal{V}}$ denote times at which miner i receives respectively $\mathcal{TR}_A$ and $\mathcal{TR}_V$, and $t_V^{\mathcal{A}}$ and $t_V^{\mathcal{V}}$ times at which $\mathcal{V}$ receives these transactions. The double-spending scenario has 3 necessary conditions $(R1, R2, R3)$ [6], which can be seen as a synchronization problem:

- $R1$: $\mathcal{V}$ adds $\mathcal{TR}_V$ in his wallet.
- $R2$: Miners confirm $\mathcal{TR}_A$.
- $R3$: $\mathcal{A}$ gets the service from $\mathcal{V}$ before $\mathcal{V}$ detects the malicious behavior.

A transaction depends on many parameters such as the sender (in the input data), the receiver (output), BTC, the time of reception, etc. In abstract setting, let us assume that a block of a blockchain does not exceed n number of transactions and in the miner's mempool there is a set of transactions $\{x_p\}_{p=1}^n, x_p \in \mathbb{R}^d$. Each miner will perform a step of similarity search, which refers to finding conflicted transactions, where he will check the format of the transactions and compare pairs of transactions $\{x_p, x_q, S_{pq}\}$, where S_{pq} represents the similarity level of $\{x_p, x_q\}$ according to some criteria. Let $P_i : \mathbb{R}^d \to \mathbb{R}$, with $P_i(x_p) = x_{p,i}$ the projection on the coordinates of x_p. According to the conditions $(R1, R2, R3)$ of a double-spending scenario defined above, two transactions are compared as follows :

$$\begin{cases} P_1(x_p) = \text{input}, P_2(x_p) = \text{output}, P_3(x_p) = \text{BTC coin.} \\ \text{Double-spending scenario of two transactions } x_p, x_q\text{:} \\ P_1(x_p) = P_1(x_q) \text{ and } P_2(x_p) \neq P_2(x_q) \text{ and} \\ P_3(x_p) = P_3(x_q). \end{cases}$$

We define $s_{pq} = 1$ if the previous conditions hold, otherwise $s_{pq} = 0$.

The similarity search consists in computing the similarity between transactions which can be represented by a matrix of similarity $(S_{pq})_{1 \leq p,q \leq n}$. If a similarity has been detected, the miner alerts the network and removes the malicious transaction. We present here how this step can be incorporated in the proof of work and how it could affect mining.

- Each miner has to wait to collect a fixed number of transactions that he will add to his block, and proceeds first to verification's steps and then, computes the similarity matrix to detect a possible malicious transaction.
- Broadcast the block of transactions and its associated similarity search and associated elapsed time.
- The time of execution of similarity search should be approximately the same from one miner to another, certainly they do not receive the transactions at the same time but they all wait a certain time to collect them and check if everything is correct.
- Each miner must notify those who put the same waiting time for similarity search and have the same block of transactions as him. This notification is considered as a favorable agreement.
- If miner's transactions are different from another miner no notifications are sent.
- The more favorable agreements the miners have, the easier the proof of work will be since its difficulty decreases according to the number of notifications and therefore it will be quick to post the new block and stop the other proofs of work that are in progress by others who have not received many notification agreements.
- The miner generates a new block which will contain an additional data as Merkle tree of the hash of the bitcoin addresses that have sent him a favorable notification.
- The miner who will succeed PoW broadcasts the following information : the block, the notifications addresses and his difficulty mining.

The similarity search of conflicting transactions prevent the synchronization problem of a double-spending scenario. where miners wait to collect a set of data in order to mine on the correct transactions. Once the similarity search is done and miners recognize if they have the same block according to the notifications, it would be interesting to adjust the difficulty of mining for those who have a large number of miners who share the same block as them. Since the mining is the brute force search to find the nonce and this is where the most energy of PoW is consumed.

It is interesting to analyze the impact of notifications on the difficulty mining. If we decrease the difficulty, the set

of possible solutions increases. So a natural question arise: how does this impact the brute force of searching for the nonce. To answer this question, a first step that we explore in this direction is an experimental study to see if there are correlations between the number of search iterations involved in the energy consumption and the difficulty mining. For that, we first recall the technical background of the difficulty mining in the Bitcoin, which depends on two value, the target and the difficulty.

Let $sha256$ be the hash function that generates a 256-bit (32-byte) number in an almost unique way, and $MRK(x_1, x_2, ..., x_p)$ the Merkle tree of the transactions. To generate a block in Bitcoin, miners work independently on solving the following problem [6]:

$$\begin{cases} \text{Find a nonce (32-bit number) such that:} \\ sha256(Bh||MRK(x_1, x_2, ..., x_p)||Nonce) \leq T, \end{cases} \quad (1)$$

where Bh is the hash of the last generated block and T is a fixed target which represents 256-bit number. This inequality fixes an interval of admitted solutions in the following sense: each miner performs a brute force search of nonce which, when concatenated with Bh and $MRK(x_1, x_2, ..., x_p)$ forms a character string S, such that $sha256(S) \leq T$. Miners rely on the hashcash [8] algorithm to find the nonce, where a starting nonce is tested in (1) and then incremented after each test until the inequality is satisfied. If all the miners start with the same nonce, it is considered the same duplicated work and therefore wasted, and the first one who present the nonce will be rewarded. To avoid this situation, in bitcoin the nonce is composed of two values: a randomized nonce different from one miner to another seen as a starting point to launch the hashcash and the nonce that will be incremented each time the inequality (1) test fails.

The larger the target is, the easier the PoW problem will be, since the solutions set will be large, and this is where the notion of difficulty mining takes place. It measures how hard it is to find the nonce, in other words how hard it is to mine a block. It has the following expression:

$$D = \frac{Target1}{T},$$

where $Target1$ is the maximum target used in the beginning of the bitcoin associated with the lowest difficulty (difficulty 1) and T (current target) is the number of solutions for which $sha256$ are less or equal T, as the difficulty increases, the set of possible solutions becomes smaller, which makes mining difficult.

In our model, we propose that this difficulty is adjusted according to the number of agreement notifications that the miner receives. The question now is how to adjust this difficulty so that the block mining is easy for a group of miners and difficult for other group of miners. For this purpose, we consider an experimental study in the following section to explore the impact of the variation of the difficulty of the mining on the number of increments performed to find a good candidate for the inequality.

IV. Experiments

By definition, if the difficulty increases, the space of candidates that solve (1) decreases, and since the hash function returns a number between 0 and 2^{256}, we expect that the total number of iterations (less or equal 2^{32}) to find the nonce increases. In this context, we are interested in the behavior of this growth of total iterations needed to test inequality (1). The experimental study will give us a first overview of this behavior, to understand how to vary the difficulty for different groups of miners and therefore impact the mining process to decrease the number of iterations.

We consider the following parameters involved in the mining process in simplified setting :

- Number of all possible nonce is 2^{32},
- Difficulty value : $2^p, p \in \{1, ..., N\}$,
- Target =2^{256-p}.

Clearly for the chosen difficulty the larger p is, the more the number of hash that satisfies (1) decreases. We test PoW by initializing the nonce by 3 values $Nonce \in \{0, 10000, 250000\}$ and incremented by one unit each time inequality (1) is tested. For fixed difficulty (fixed p), in each iteration, where nonce is incremented, we test $sha256(Bh||MRK(x_1, x_2, ..., x_p)||Nonce) \leq Target$ until it is satisfied, we then consider the total iterations performed to find the good candidate associated with the fixed difficulty. Implementation is available at https://github.com/Lydiaouaili/difficulty_mining_bitcoin. In (1), we replace $Bh||MRK(x_1, x_2, ..., x_p)||$ by a simple character string 'expression', we test $sha256('expression'||Nonce) \leq 2^{256-p}$ for a fixed $p \in \{1, ..., 30\}$ and we increment $Nonce = Nonce + 1$, until the right candidate is found. These steps are performed for different difficulty values, which depend on p. We are interested in the the increasing behavior of the necessary iterations to find the candidate for each difficulty.

The following figures shows the number of iterations of finding the nonce in the PoW for a fixed difficulty with different starting nonce:

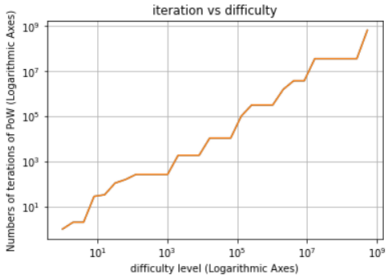

Fig. 1. Total number of iterations performed for each fixed difficulty in PoW with starting nonce=0

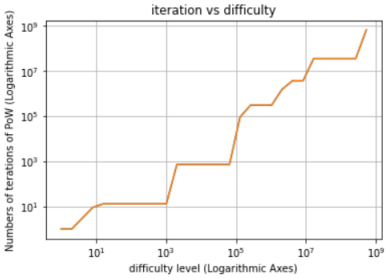

Fig. 2. Total number of iterations performed for each difficulty in PoW with starting nonce=10000

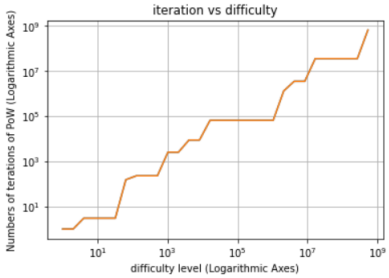

Fig. 3. Total number of iterations performed for each difficulty in PoW with starting nonce=250000

The previous figures can be explained in intuitive way as follow:

- A high difficulty means a restricted space of hash results, which increases the number of iterations to find the nonce, so we notice an increasing behavior between the difficulty and the total number of iterations.
- A a non-linear behavior and phases of stagnation are observed, where even if the difficulty increases we can perform almost the same iterations to find the right candidates. There are phases where the iterations are constant. This can be explained by the randomness and unstructured nature of the hash function : the output of hash function must be uncorrelated from the input.

Indeed by changing the difficulty to reduce the set of possible solutions of the nonce, the PoW becomes difficult, but recall that, our goal is to minimize the total iterations by adjusting this difficulty, where each miner will have a difficulty that depends on the number of miners who have the same block as him. A first naive idea is to reduce the difficulty, but according to the experimental results an interesting behaviors appear. From the previous remarks the gap between the difficulties must be big to reduce significantly the number of iterations. If the gap between difficulties is small it may not decrease the iterations, we observe a phenomena of iteration constancy.

V. Conclusion and Futures Works

In this paper, we focused on the possibility of reducing the energy consumed during the mining process, one of the most crucial step of the PoW consensus, used in many permissionless blockchains such as bitcoin. We propose to reduce this energy by acting on the difficulty mining during the generation of a block, first by removing conflicting transactions before mining to prevent the attack, which is part of the conditions necessary to secure the blockchain of bitcoin. Then we focused on the adjustment of the difficulty in order to reduce total iterations performed in the mining. The experimental results lead us to the following future works:

- Model the link between the difficulty and the number of iterations to formalize the observed non-linear behavior.
- Formalize the necessary gap between different difficulties to modify the number of iterations and reduce them.
- The experiments can be improved by comparing with related work. In cryptography, interesting work [13] focuses on the optimization of the hash function and the possibility of finding a faster method in the mining process than the brute force search. In the context of energy, it would be interesting to study their energy impact.

References

[1] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 2008. p. 21260.
[2] Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 2018. 14(4), 352-375.
[3] O'Dwyer KJ, Malone D. Bitcoin mining and its energy footprint. 2014 In: 25th IET Irish signals & systems conference 2014, pp 280–285
[4] Sedlmeir, J., Buhl, H. U., Fridgen, G., & Keller, R. The energy consumption of blockchain technology: beyond myth. Business & Information Systems Engineering, 2020, 62(6), 599-608.
[5] Platt, M., Sedlmeir, J., Platt, D., Tasca, P., Xu, J., Vadgama, N., & Ibañez, J. I. Energy footprint of blockchain consensus mechanisms beyond proof-of-work. (2021) arXiv preprint arXiv:2109.03667, unpublished.
[6] Karame, G. O., Androulaki, E., & Capkun, S. Double-spending fast payments in bitcoin. In Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 906-917.
[7] Podolanko, J. P., Ming, J., & Wright, M. Countering double-spend attacks on bitcoin fast-pay transactions. In Proc. Workshop Technol. Consum. Protection, 2017, pp. 1-3.
[8] https://en.bitcoin.it/wiki/Hashcash.
[9] Küfeoğlu, S., & Özkuran, M. Bitcoin mining: A global review of energy and power demand. Energy Research & Social Science, 2019, 58, 101273.
[10] Lasla, N., Alsahan, L., Abdallah, M., & Younis, M. Green-PoW: An Energy-Efficient Blockchain Proof-of-Work Consensus Algorithm. (2020) arXiv preprint arXiv:2007.04086, unpublished.
[11] Zhang, F. Eyal, I. Escriva, R. Juels, A., & Van Renesse, R. REM: resource-efficient mining for blockchains. In Proceedings of the 26th USENIX Conference on Security Symposium (SEC'17). USENIX Association, USA,2017, 1427–1444.
[12] Daian, P., Eyal, I., Juels, A., & Sirer, E. G. (Short Paper) piecework: Generalized outsourcing control for proofs of work. In International Conference on Financial Cryptography and Data Security 2017 (pp. 182-190). Springer, Cham.
[13] Courtois, N. T., Grajek, M., & Naik, R. (2014, September). Optimizing sha256 in bitcoin mining. In International Conference on Cryptography and Security Systems (pp. 131-144). Springer, Berlin, Heidelberg.