mathematics

MDPI

Article

Neural Fairness Blockchain Protocol Using an Elliptic Curves Lottery [†]

Fabio Caldarola [1,*], Gianfranco d'Atri [1] and Enrico Zanardo [2]

[1] Department of Mathematics and Computer Science, Cubo 31/A, Università della Calabria, 87036 Rende, Italy
[2] Department of Digital Innovation, University of Nicosia, 46 Makedonitissas Avenue, CY-2417, Nicosia P.O. Box 24005, Cyprus
[*] Correspondence: caldarola@mat.unical.it

[†] Dedication: This article is dedicated to the memory of an exceptional person, Dr. Giovanni Bozzolo. He graduated in mathematics from our university and was thrilled to teach this subject. He tragically passed away at the age of 27 in an accident two days before Christmas 2021.

Abstract: To protect participants' confidentiality, blockchains can be outfitted with anonymization methods. Observations of the underlying network traffic can identify the author of a transaction request, although these mechanisms often only consider the abstraction layer of blockchains. Previous systems either give topological confidentiality that may be compromised by an attacker in control of a large number of nodes, or provide strong cryptographic confidentiality but are so inefficient as to be practically unusable. In addition, there is no flexible mechanism to swap confidentiality for efficiency in order to accommodate practical demands. We propose a novel approach, the neural fairness protocol, which is a blockchain-based distributed ledger secured using neural networks and machine learning algorithms, enabling permissionless participation in the process of transition validation while concurrently providing strong assurance about the correct functioning of the entire network. Using cryptography and a custom implementation of elliptic curves, the protocol is designed to ensure the confidentiality of each transaction phase and peer-to-peer data exchange.

Keywords: blockchain; distributed consensus; neural networks; elliptic cryptographic curves; decentralized applications; tokens; machine learning; confidentiality preserving; cryptocurrencies

MSC: 68M1; 11T71; 68T07

check for updates

Citation: Caldarola, F.; d'Atri, G.; Zanardo, E. Neural Fairness Blockchain Protocol Using an Elliptic Curves Lottery. *Mathematics* **2022**, *10*, 3040. https://doi.org/10.3390/math10173040

Academic Editor: Jan Lansky

Received: 2 July 2022
Accepted: 17 August 2022
Published: 23 August 2022

1. Introduction

Over the past decade, the advent of distributed systems combined with cryptography and the success of Bitcoin, based on Blockchain technology, demonstrates the potential and interest in this new way of interaction and exchange. Trust has been a point of discussion in limited circles of enthusiasts for decades before being formalized for the first time by D. Chaum in [1]. Between then and the introduction of Bitcoin [2] in 2008, numerous researchers [1,3–6] have studied the elimination of intermediate parties and direct communication between stakeholders, which allows for an increase in trust through the replication of information and validation processes in a network.

The first mainstream occurrence was the release of the Bitcoin whitepaper [2], which unleashed a new era in this research field.

After the introduction of the genesis Proof-of-Work consensus protocol [7,8] for Bitcoin, many others have been introduced for blockchain such as the following:

(1) *Proof-of-Stake* (PoS) [9] replaces Proof-of-Work-based mining with a mechanism that makes the chances of mining a block proportional to the amount of stake (currency) a miner has.
(2) *Proof-of-Authority* (PoA) [10] is a regulated consensus mechanism that uses authorized nodes or validators to obtain consensus. Validators are limited to n authorized sealers. This protocol works by validating machines to produce network transaction blocks.

Each blockchain network's architecture determines block acceptance and mining. In this procedure, the sealer's reputation is at stake; therefore, hostile behavior is harmful to the organization.

(3) *Proof-of-Accuracy* (PoAC) [11] aims to democratize the miners' participation within a blockchain, control the miners' computing power, and mitigate the majority of attacks.

(4) *Proof-of-Capacity* (PoC) [12] allows for mining devices in the network to use their available hard drive space to decide mining rights and validate transactions.

They are the consensus protocols in the top 50 cryptocurrencies by market capitalization from CoinMarketCap [13]. For overviews of literature, alternative protocols, and specialized papers on various aspects, the reader can see [14–18].

The Bitcoin and Ethereum public chains only provide minimal anonymity for user identities, and transaction amounts are visible to the whole network, resulting in a breach of user confidentiality. Based on the current anonymous technology, the sender, recipient, and transaction amount are hidden and no information is disclosed, making oversight impossible. Therefore, the design of a blockchain system with confidentiality and monitoring capabilities is of critical importance. Increasingly, the necessity to protect the confidentiality of the users participating in a transaction is a basic requirement that must be considered during each phase of a data transaction between a client (also known as a wallet) and a distributed ledger.

1.1. Research Problem

Blockchain is a revolutionary technology that prioritizes the secure transmission of data between many distributed applications. Despite its broad usage, there are still areas in which further study is necessary to comprehend its performance characteristics. Moreover, consensus algorithms, a crucial component of the blockchain, demand a deeper comprehension of its underlying concepts and technical qualities. Several issues, such as system scalability and energy usage, have been highlighted in conjunction with the development of various consensus methods. Therefore, more research is required to examine the degree of construction and performance of consensus algorithms. We present a novel consensus algorithm that prioritizes confidentiality above blockchain performance, with the purpose of boosting safety without affecting blockchain behavior.

1.2. Our Contributions

Our contributions include the development of a novel Fairness Consensus Protocol that protects the confidentiality of the user's data and enables the network to reach consensus using a combination of neural networks, cryptography, and conflict graphs. They will be interconnected with a personal version of Delegated Proof-of-Stake protocol [19] that will use a dedicated crypto-lottery to elect the leader and the committee. They will decide which transactions will be stored into the distributed ledger. Preventing the same nodes from forming the consensus committee could increase network fairness. We aim to obtain a fair consensus as quickly as possible, avoiding a scenario where only a few nodes have 51 percent consensus power. Fair consensus is when all honest nodes have equal odds of making the final board. All trustworthy nodes do not contribute evenly to the network. Not all nodes can retain the whole block history in memory, and not all are available 24/7. It is tough to pick a group of nodes with specific qualities that can assure fast block production. Focusing primarily on this utility may disrupt block validation boards' balance. The Conflict Graph provides a solution. Using only utility could lead to an unbalanced Board with the same nodes. A Conflict Graph prevents this, maintaining fairness.

1.3. The Paper Structure

We briefly describe the structure of this paper here. In the next section, we will define the fairness consensus protocol and reserve four paragraphs for the main components, namely, weighted nodes, consensus committee, verifiable cryptographic lottery, and neural network model. Section 3 details the operational logic of the protocol, outlining, in particular, six phases. Section 4 is dedicated to the board election algorithm and the role of elliptic curves. It is divided into nine subsections that examine the details (utility factor, conflict of interest, elliptic curves framework, tickets generator, leader selection algorithm, etc.). Finally, Section 5 briefly discusses rewards and Section 6 concludes the paper with some future work.

2. Abstract Definition of the Protocol

This paper presents a new fairness consensus protocol designed to confirm transactions while preserving the confidentiality of the users and of their own data that are transmitted into the network. The core of this system uses a Byzantine agreement protocol that scales to many users, which allows the network to reach consensus on a new block fast and with low latency. A key technique that makes this protocol very efficient and fair is the use of verifiable lottery functions to randomly select users in a private and noninteractive way and the use of a neural network model that is updated and distributed to all the nodes each time a consensus is reached. The Fairness Consensus Protocol addresses these challenges using several techniques, as follows:

Weighted nodes. To prevent Sybil attacks, the neural network assigns a weight to each node. The protocol is designed to guarantee consensus as long as a weighted fraction (a constant greater than 2/3) of the nodes are honest. The weight of the nodes is based on the score that the neural network calculate each time a node is elected to be part of the committee and vote for a set of transactions.

Consensus committee. Similar to the Delegated Proof of Stake, the system achieves scalability by choosing a committee comprising a subset of nodes that, through a crypto-lottery, are randomly selected from the total set of nodes. From the verifiable lottery, which is run independently in each node, each node will be able to know if it will be the winner of one place into the committee. The first place of the lottery is designed for the leader. The leader is the node that will start the consensus phase proposing a set of transactions that could be written into the distributed ledger. All other nodes observe and broadcast the protocol messages, which allows those that are part of the committee to agree upon the proposed set of transactions.

Verifiable Cryptographic Lottery. A verifiable cryptographic lottery (VCL) selects committee nodes in a noninteractive and private manner to prevent an adversary from targeting committee members. This means that every node in the system can independently determine if it is selected for the committee by computing the verifiable function using elliptic curves and public information from the blockchain, such as the seed that is randomly generated each time a block is created. Since the verifiable cryptographic lottery is noninteractive, an adversary is unaware of which nodes are members of the committee that elects the following block.

More specifically, the VCL is based on hashing functions and elliptic curve cryptography (ECC), and works synthetically as below (see Sections 4.6–4.9 for a detailed description). In this paper, we moreover adopt the so-called Simple Uniform Hashing Assumption (SUHA), which is commonly used in theoretical computer science to allow an easier mathematical analysis of the processes (see, for example, [20–22]). Once an elliptic curve has been chosen on which to carry out the calculations (for example, one of the type Secp256k1, as characterized in [23], Section 2.4), it is possible to generate N tickets using the group law on the elliptic curve. The explicit law can obviously vary considerably by favoring structural simplicity, as we will detail in Section 4.7, or by exploiting particular constructs such as generalized Fibonacci sequences (see [24–28] and the references therein), new computational paradigms based on unimaginable numbers (see [29,30]), n-sets (see [31]), and much more. In the framework we use in this paper, in particular, a ticket is associated with a point on the elliptic curve, and in most cases, we always start from two points that determine the sequence of tickets t_i either explicitly or recursively.

Then, we consider the hash of the last block written in the blockchain and, dividing it into eight parts, compute the resulting XOR from all of them. This means calculating seven XOR (or, equivalently, a unique n-ary XOR operation) to obtain a final binary number, which will give the leader, if thought of as a number in the set $\mathbb{N}$ of naturals and taking the rest modulo N (see in particular, Section 4.8).

Neural Network model. To decide if a set of transactions could be stored into a block and distributed to all the peers, a dedicated neural network model is used. It engages confidentiality-protecting techniques such as local and global differential confidentiality and encrypted deep learning. However, each set of techniques is only useful for the trust configuration that it supports. Some techniques would be overly burdensome amongst data and model owners who trust each other implicitly, while others would be too insecure for the trust model of potentially malicious actors. This protocol is built as a cross-section of fundamental techniques that are applicable to a broad spectrum of confidentiality settings. It is useful when a party wants to annotate a local dataset using the private datasets of other actors and the epsilon-delta tool allows for very granular control of just how much the other actors must trust us to protect their confidentiality in this process. Finally, secure additive aggregation helps add additional confidentiality protections in this setting—in the latter case, preventing anyone from seeing an aggregated gradient from one individual, which is a much stronger confidentiality protection. Each time a consensus is reached, the updated version of the neural network model is distributed along the network and allows the next elected committee to be more conscious about the future proposed set of transactions. Using this new technique, the model is distributed and the data will be kept as private as possible. However, the paper does not describe any specific solution for confidentiality preserving (see, for example, [32,33] and the references therein).

3. Neural Fairness Protocol Logic

The Neural Fairness Protocol is needed to generate the new block and guarantee that the system is simultaneously **scalable**, **secure**, and **decentralized**. The NFP process that writes a block into a blockchain could be split into six subphases.

In the first phase, in a noninteractive way, each node runs a verifiable cryptographic lottery (VCL) that, starting from a public randomly generated seed, will allow all the nodes to run the same lottery and determine which nodes will be part of the committee and which one will be the leader who proposes the next block.

The Neural Fairness Protocol is derived from the Delegated Proof-of-Stake consensus; therefore, the number of tickets that will be available for the lottery is equal to the number of stakes that a node has. Similar to the Delegated Proof-of-Stake, the Neural Fairness Protocol relies on the Honest Majority of nodes. Accordingly, some members of the committee may be chosen twice or, more generally, k times; in this case, members chosen k times will have k votes in the committee to propose valid transactions for the next block. This phase reflects real world/society. Similar to the real world, a small percentage of corrupted nodes may be involved in trying to propose a set of transactions for a block.

Finally, for clarity, we specify that if the computed token belongs to a node that is not eligible for the committee, the process is repeated.

The second phase takes into account the communication of the nodes about the list of requested transactions that are waiting to be stored into a block.

The literature on distributed computing (as well as the cryptography literature) typically considers two types of players: honest players and corrupted players. Resilience properties are then analyzed, assuming a lower bound on the fraction of honest players. Honest players, however, are not only assumed to follow the prescribed protocol, but also assumed to be online throughout the whole execution of the protocol. The advent of "large-scale" consensus protocols (e.g., the blockchain protocol), where we may have millions of players, makes this assumption unrealistic [34].

Following those assumptions, the neural fairness protocol is build in a "sleepy" model of computation, where players can be either online (awake) or offline (asleep) and their online status may change at any point during the protocol execution. Therefore, nodes that join the network can be honest or corrupt.

The third phase is dedicated to an internal process that each node will run independently. Every time a node will be part of the network, it will start to receive transactions from the clients; each time a transaction is received by the node, the node will parse the transaction's features into its internal neural network model. The output of the neural network model will provide a score to the given transaction. Each node will collect and sign all of these results in order to further distribute them across the network while maintaining the confidentiality of the data contained within the parsed transactions. The communication between nodes is made secure by using a kind of Elliptic-curve-based Diffie–Hellman Key Agreement (ECDH) [35] to generate a shared symmetric key. In particular, the Ristretto one is used [36].

Next is the fourth phase. After the VCL execution, each node will know if it will be elected to be part of the committee, or better, to be the leader. Using a gossip protocol for communication propagation, the leader will start to spread its own transaction proposals to the other nodes. Nodes that receive the transaction proposal will verify that it is coming from the leader and is not corrupted. Here, the statement to be verified about the hash function's input is in the zero-knowledge, and a zero-knowledge proof-friendly hash function Poseidon is used in order to exploit its capability to minimize proof generation time, proof size, and verification time [37]. These proprieties are well-suitable for working with the proof system we use. Indeed, Bulletproofs [38] is used as proof system because the absence of a trusted setup, usually needed for the shortest-range proofs, is an important requirement in a context such as Distributed Ledger Technology and blockchain. However, Bulletproofs can be used for general verification statement or *generic* Zero-Knowledge Proofs, not only the shortest-range proofs [38].

Once the verification is passed, they will start to execute their own neural network using the data contained in the transaction proposal. Based on the outcome, they will provide their own opinion, sign, and spread the outcome to the other nodes.

In the second-last phase, the fifth, the leader will receive back all the outcomes provided by the committee. It will verify their origin and it will execute the neural network with that data.

As before, it will sign and spread the outcome to the network. The nodes of the committee will execute again the neural network and vote for the final proposal.

Finally, in the last phase, the sixth, the committee, based on the received final votes, will know if the proposed transaction will be considered valid or not. If the final result is considered valid, the committee will write the new block into the blockchain and distribute the updated version of the neural network model to the network within the new block. If a node becomes corrupt, the stake under the control of the particular node is considered to be corrupt. A node is defined as an honest node if it does not deviate from the prescribed protocol execution rules.

Every time a final decision is taken, the process will restart from the first phase.

4. Elliptic Curves and the Board Election Algorithm

This section is devoted to a more in-depth description of how the cryptographic lottery exploits the arithmetic properties of elliptic curves to determine the board committee, which is of central importance in the proposed consensus procedure. First of all, it is possible/convenient to consider a restricted subset of the network nodes according to their (i) minimum balance, (ii) utility factor, and (iii) conflict of interest (see Sections 4.1–4.5).

A cryptographic lottery (CL) is used to randomly and fairly select committee members. It is based on the use of elliptic curves: the lottery output is an integer n, which indicates the corresponding multiple nP of a point belonging to a certain fixed cryptographic elliptic curve whose group law is considered, as usual. Thus, the nP point obtained identifies a wallet—the owner of the specified token. In the following, we will describe the procedure in four distinct steps (see Sections 4.6–4.9).

Instead of elliptic curves, it is also possible to use schemes based on some types of conics, for example, the Pell hyperbola (see [39]), and thus obtain some advantages such as a higher system speed.

4.1. Minimum Balance

A minimum balance $B \geq \rho$, where ρ is a token quantity to be defined, is a requirement for membership in the committee. All nodes with a balance of at least B have a chance to participate in the Consensus method if and only if a utility factor is assigned to each of them. In addition, the minimum required balance must be available for a specific period of time in order to be considered. The difference between a minimum balance accessible today and a minimum balance available in 30 days is significant in order to calculate the final score of each node.

4.2. Utility Factor

The Utility Factor is a metric that determines how useful a node is to the entire network, and thus, how much a user is entitled to be a member of the committee. A committee member will participate in the consensus process and, as a result, in the construction of the new block. This is a critical task that must be completed by as many trustworthy nodes as feasible.

4.2.1. Utility Factor Definition

The technique for assigning a utility factor is similar to assigning a value of importance to a network node. The network under examination is the direct graph of transactions, with the set V of vertices representing nodes in the network that exchange transactions and the set E of edges; $(i, j) \in E$ if and only if at least a transaction from the node A_i to the node A_j occurs. Everyone has access to transaction information, and the consensus mechanism controls the dynamic of validated transactions. This means that a transaction must be validated by the network through a distributed consensus in order to be included in the transaction graph. As a result, using the graph transaction as the reference graph to determine each node's usefulness ensures the following:

1. Anyone can check if the utility associated with a node is correct by computing the value using public open data as transactions;
2. Information contained in the transaction graph is not malleable as long as the consensus procedure is secure;
3. The transaction graph evolves over time, ensuring that new information about a node's behavior is taken into account.

For assigning an importance value to a node in a network, there exist some outstanding algorithms. One difficulty is determining a node's centrality in the network by computing its centrality degree. The PageRank method [40], which obtains the rank of a web page as the output of an iterative linear process applied to a Markov chain transition probability matrix, is one of the most well-known in the context of the Web. The Proof of Interest (PoI) is based on the NEM project, specifically, the outlink matrix below, which is part of the NEM PoI protocol.

4.2.2. The Outlink Matrix

Suppose the PoI calculation is done at height h. The protocol considers, for each node with a balance $b \geq B$, all its transactions with the following properties:

- Transferred an amount of at least ϕ tokens;
- Happened within the last 43k blocks (approximately 30 days).

For each such transaction T_k that transferred an amount θ from the account A_i to the account A_j and happened at height h_{ijk}, a weight is associated to this transaction as follows:

$$w_{ijk} = \theta \cdot \exp\left(\log(0.9) \left\lfloor \frac{h - h_{ijk}}{1440} \right\rfloor \right), \tag{1}$$

where $\lfloor \cdot \rfloor$ denotes the floor function. The weight is obtained as the output of an exponential function depending on the time; so, a transaction that happened today has a higher associated weight than a transaction that happened in the last 30 days. Summing up all transactions

$$\hat{w}_{ij} = \sum_k w_{ijk} \tag{2}$$

and setting

$$\hat{o}_{ij} = \begin{cases} \hat{w}_{ij} - \hat{w}_{ji} & \text{if } \hat{w}_{ij} - \hat{w}_{ji} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

finally, the outlink matrix O is obtained as follows:

$$o_{ij} = \begin{cases} \dfrac{\hat{o}_{ij}}{\sum_i \hat{o}_{ij}} & \text{if } \sum_i \hat{o}_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The outlink matrix element o_{ij} gives the **weighted net flow** of tokens from A_i to A_j during the (approximately) last 30 days. In summary, the outlink matrix takes into account only the positive net transactions flow between nodes.

4.2.3. Utility Formalization

The protocol seeks to categorize stakeholders' behavior in terms of their activities in the network layer, in addition to taking into consideration their behavior within the transaction graph. A final score is given in accordance with particular network layer features, which will be added to the utility as stated up to this point. The key goal is to identify stakeholders who have a lot of network activity, a lot of round-trip time, and a lot of throughput. Imposing a greater utility factor on stakeholders who demonstrate higher reliability in the network layer, allows us to move closer to the synchronous network assumption, which allows us to specify an upper constraint on message delays. This assumption enables for real-time operations, as well as various theoretical benefits for the consensus. Furthermore, requiring the committee to be formed by stakeholders who are as accessible as they are feasible, provides for a close approximation of the premise of having a predetermined number of stakeholders available in the committee when consensus is needed.

The features considered are listed in Table 1. They are all gathered for a set period of time, termed an era. For some, history is gathered, while for others, it is refreshed in each era.

Table 1. Features.

Features	**Values**
Total number of TCP connection	integer
Sum of all uploaded bytes	float
Sum of all downloaded bytes	float
Sum of the number of retransmitted bytes from peer	float
Average round trip time between peers	float
Average time for processing and returning the first segment with payload since the first flow	float
Ping	float

The abovementioned features define the utility, implying that a specific configuration of these traits identifies the stakeholder to whom the best utility must be assigned.

Now, we will look at two strategies for calculating utility. The first provides a basic scenario for examining the problem without utilizing a computationally intensive strategy, whereas the second is a more powerful learning algorithm that can achieve better results but requires more computational resources.

4.3. Baseline Utility Algorithm

The system chooses the best and worst stakeholder's feature configurations, which identify the maximum and minimum utility, respectively. A *Self-Organizing-Map* (SOM) is trained to identify the *Best Unit Match* (BUM) neurons on the map from these two selected attributes, which we will refer to as *targets utility stakeholders* (TUSs). Following that, all stakeholders' samples are mapped to the trained grid, where each is assigned to a BUM neuron in the mapping step. A Euclidean distance is computed between the weights associated with the BUM's TUSs ($w_{\text{target}\,1}$, $w_{\text{target}\,2}$) and the BUM's weight w allocated to a stakeholder, and a utility is assigned to the stakeholders based on these two distances:

$$U = \begin{cases} 1 + \exp(-d_2) & \text{if } d_1 > d_2, \\ 1 - \exp(-d_1) & \text{if } d_2 > d_1. \end{cases} \tag{5}$$

The pseudocode of the baseline utility algorithm is reported in Algorithm 1:

Algorithm 1 Baseline Utility algorithm pseudocode

```
Input: Two targets nodes and a subset of samples nodes.
Output: A list of utility values to be assigned to the sample nodes
while $Node\ in\ Nodes$ do ▷ train the SOM with the two target samples nodes and obtain targets weights
    predict the BMU;
    obtain weights $w$ associated to the predicted BMU;
    compute distance $d_1$ between $w$ and $w_{\text{target}\,1}$;
    compute distance $d_2$ between $w$ and $w_{\text{target}\,2}$;
    if $d_1 > d_2$ then
        $u \leftarrow 1 + \exp(-d_2)$
    else
        $u \leftarrow 1 - \exp(-d_1)$
    end if
end while
```

The utility u is a value in the range $[0,2] \subset \mathbb{R}$. If a stakeholder is connected with the poorest target neuron, its utility is in the range $[0,1]$, but if it is associated with the best target neuron, it is in the range $[1,2]$, based on the distance between its weights and the target neuron's weights.

4.4. Reinforcement Learning Utility Algorithm

There are different approaches to calculate the utility factor, as previously discussed, and the Deep-Q-Network model is given in this paragraph to assign the nodes an appropriate utility. The Q-learning method is presented in detail, as well as how to solve a Reinforcement Learning problem using the well-known supervised Deep Neural Network theory.

4.4.1. Markov Decision Process (MDP)

Reinforcement MDP is inextricably linked to learning [41]. MDP is a stochastic discrete-time control process. It gives a mathematical framework for modeling decision-making in settings where outcomes are partially random and partly controlled by a decision maker. The following are the primary components of MDP:

- Environment;
- Agent, a decision-maker interacting with the environment performing subsequent actions;

– States, representations of the environment under certain conditions;
– Action, performed by the agent with respect to the state;
– Reward, consequence of the action given to the agent.

Given a finite set of states S, a finite set of actions A, and a finite set of rewards R, at each time step $t = 0, 1, 2, \ldots$, the agent receives some representation of the environment's state $s_t \in S$. Based on this state, the agent selects an action $a_t \in A$ resulting in the state–action pair (s_t, a_t). Time is then incremented to the next time step $t + 1$, and the environment transits to a new state s_{t+1}. At this time, the agent receives a numerical reward $r_{t+1} \in R$ for the action performed in the state s_t. The reward assignment is represented by a function $f(s_t, a_t) = r_{t+1}$.

4.4.2. Deep-Q-Network Utility Model

The DQN utility model is described below. First and foremost, states, actions, and rewards must all be defined. The Deep-Q-Network architecture, as well as its hyperparameters and metrics, are then described.

Nodes (agents) must choose an action for each state in this scenario. States are samples that report some characteristics of each node's behavior. Actions reflect how much of a node's right to participate in the final consensus is determined. There are a few things to think about in general.

An initial training phase is required for a Reinforcement Learning (RF) model, in which agents learn the optimum method to execute actions in order to maximize a reward signal function. Errors in an Atari game meant losing the game; in a blockchain system, they may mean losing stakes. As a result, it was decided to train the network in a TestNet phase, during which users can join the network and conduct transactions using TestNet Coins. Users must choose one of the following behavior scenarios at the start of their connection at this phase:

1. Honest Player (HP): the user joins the network with the intention of behaving as an honest player, exchanging transactions with other users, while never acting in a malicious way.
2. Bad Player (BP): the user joins the network for making attacks to the network.

During the TestNet phase, each node (agent) collects information about other nodes in a noninteractive manner (agents). The states of the environment are made up of these characteristics. If the network does not converge to a defined threshold, detailed later, features may alter during the TestNet phase. The following are the initial features:

– UpTime, the total time a node results to be connected to the network.
– OutLink Matrix, as described in Section 4.2.2.
– Vested Balance, the node's total balance in times, defined as follows:

$$\rho = \beta \cdot \exp(T) \tag{6}$$

where β is the current balance and T is the total time that such balance is present.
– Total number of TCP connection (TTCP).

From the features, a state is defined as a $1D$ tensor of dimension 5, as follows:

$$s = [\text{UpTime, OutLink Matrix, Vested Balance, TTCP, Behaviour}]$$

where *Behaviour*, is the selected behavior scenario each node must choose at the beginning: 1 for HP, 0 for BP.

Agents must select a specific action from a list of options. While states can change, all TestNet phases have the same actions. The following is the list:

1. Participate with 100% tickets: 0;
2. Participate with 75% tickets: 1;
3. Participate with 50% tickets: 2;

4. Participate with 25% tickets: 3;
5. Participate with 0% tickets: 4.

Agents are compensated for their efforts. At first, the system chooses two states to represent the best (sBest) and worst (sWorst) behaviors that a node can exhibit. The awards are calculated using these states. The following are all conceivable scenarios: for each state, an agent must choose an action.

– If the current state belongs to a HP ($Behaviour == 1$), an agent should choose an action in the set $\{0, 1, 2\}$. In particular, a Euclidean distance is computed between the given state and the sBest. If the distance is greater than 2, and the agent has chosen the action 1 or 2, it receives a reward of $+1$; otherwise, it receives 0 rewards. If the distance is less than or equal to 2, if the agent has chosen the action 0, it receives a reward of $+1$; otherwise, it receives 0 rewards.
– If the current state belongs to a BP ($Behaviour == 0$), an agent should choose an action in the set $\{3, 4\}$. In particular, a Euclidean distance is computed between the given state and the sWorst. If the distance is greater than 2, and the agent has chosen the action 3, it receives a reward of $+1$; otherwise, it receives 0 rewards. If the distance is less than or equal to 2, if the agent has chosen the action 4, it receives a reward of $+1$; otherwise, it receives 0 rewards.

The Behavior feature is only utilized in the TestNet phase; once the system converges and a MainNet phase can begin, the Behavior feature will be removed, but the network will have learned how to take appropriate action given the states indicated by the selected features. As the network has already been trained, nodes with the DQN's weights will be able to predict the correct action given a state in that phase and incentives will be unnecessary.

4.5. Conflict of Interest (CoI)

It could be useful, for boosting network fairness, to avoid having the same nodes form the consensus committee all of the time. In fact, its use is fairly broad and adaptable to a variety of situations that arise in a distributed consensus method. We want to obtain a fair consensus as quickly as possible, avoiding a situation where only a few nodes have a consensus power equal to or more than 51 percent. When all honest nodes have an equal chance of being on the final board, the consensus is said to be fair. Even if all nodes are honest, they do not all have the same usefulness in the network. For example, not all nodes can cache all of the block history in memory, and not all nodes wish to stay up to 24 h a day. As a result, the difficulty of identifying a subset of nodes with specified characteristics capable of ensuring the fastest possible block production arises. However, focusing solely on such a utility may throw off the balance of the boards designed to validate blocks. The notion of Conflict Graph is introduced to overcome this problem.

Conflict Graph (CG). Relying only on the utility factor could lead to a situation of imbalance in which the same nodes will always be part of the final Board. In order to avoid this, and therefore, guarantee a level of fairness as high as possible, a Conflict Graph is used. A *Conflict Graph G* is defined as a pair (V, E), where V is the set of nodes and E the set of edges. Edges $e_{i,j}$ or (i, j) between the node i and the node j exist if these nodes may have a Conflict of Interest (CoI). A CoI exists when two nodes could have either some interests in common or they have experienced some event together.

With this in mind, the mathematical model that outputs which nodes could be part of the next board is as follows:

$$\max \sum_{i=0}^{N} U(a_i) \cdot x_i \tag{7}$$

$$\text{s.t. } x_i + x_j \leq 1 \text{ if } (i, j) \in E_G \tag{8}$$

where

– $U(a_i)$ is the utility associated to the node a_i;

- $x_i \in \{0, 1\}, i = 1, \ldots, N$, where N is the total number of nodes for which an utility exists;
- G is the Conflict Graph, composed by the binary decision variables, and E_G is its set of edges.

Such problem is known in literature as the Maximum Weight Independent Set [42]. This is a well-known NP-hard problem, and several possible solutions and algorithms have been proposed. Mari [43] and then Chaudhry [44] proposed an intuitive and straightforward greedy algorithm, which, at each iteration, adds the minimum degree vertex to the solution and deletes the neighbors from the solution space. Goldberg and Spencer [45] gave a parallel algorithm for MIS, which found an independent set with a size of at least $n/(d_G + 1)$, where d_G is the degree of G and n is the total number of its nodes. If a maximum total weight is considered, the problem reduces to the disjunctively constrained knapsack problem, which can be solved with the heuristic proposed in [46].

The basic principle behind the primary method is that for each subnetwork described as the conflict graph that a node defines with its conflict nodes, the node with the highest weight is chosen as the solution, while the others are eliminated.

The algorithm explores all the graph node by node. For a given node i, it computes the i's conflict nodes, in order to compute a list with the weights of each of them, plus the weight of the node i. From this list, the maximum value is selected, and the node with this weight value is chosen as solution, while all others are deleted from the initial graph.

The algorithm solves the Maximum Weight Independent Set (MWIS) in polynomial time rather than exponential time; however, it achieves less accuracy than the best case of around 40% of the best response, which is based on a recursive approach to the problem.

The pseudocode of the Algorithm 2 is as follows:

Algorithm 2 MWIS pseudocode

```
  Input: A weight graph G.
  Output: A maximum independent subset of G
conflictNodes = 0;                                              ▷ initialize
while $n \in G$ do
    $conflictNodes \leftarrow N_G(i)$;
    $totalUtility \leftarrow w(N_G(i)) + w(i)$;
    select the maximum weight;
    select the corresponding node that has, as its weight, the maximum value;
    delete from $G$ all other nodes;
end while
```

The CoI method is a service that can be tailored to the specific environment in which the blockchain system is implemented. The meaning of the term "conflict of interest" varies depending on the context of the application. In an application where Smart Contracts are traded, for example, a conflict of interest can occur if members of the committee are affiliated with the entity proposing the same smart contract. In this situation, a clear personal interest can affect the judgment of the proposing smart contract, avoiding an impersonal review.

4.6. The Framework with Elliptic Curves

We choose an Elliptic Cryptography Curve (ECC) that originates a cryptographic system specified by a sextuple of parameters (see [23]),

$$T = (p, a, b, G, n, h),$$

where p is a prime number that provides the finite field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, and $a, b \in \mathbb{F}_p$ are parameters specifying an elliptic curve

$$E = E(\mathbb{F}_p): \quad y^2 \equiv x^3 + ax + b \pmod{p},$$

$G = (x_G, y_G)$ is a base point on E, n is a prime denoting the order of G, and h is an integer representing the cofactor $h = |E(\mathbb{F}_p)|/n$.

The `secp256k1` is a 256-bit elliptic curve with parametric sextuple T as follows.: ut is defined by $y^2 = x^3 + 7$ over the field $\mathbb{F}_p$, where

$$\begin{aligned} p &= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \\ &= 115\,792\,089\,237\,316\,195\,423\,570\,985\,008\,687\,907\,853 \\ &\quad\ 269\,984\,665\,640\,564\,039\,457\,584\,007\,908\,834\,671\,663 \\ &= \text{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad\ \text{FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F} \end{aligned}$$

is a special form prime number with 78 decimal (and 64 hexadecimal) digits. The base point G, in its compressed form, is

$$G = \text{02 79BE667E F9DCBBAC 59F2815B 16F81798},$$

the order of G is

$$\begin{aligned} n &= \text{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE} \\ &\quad\ \text{BAAEDCE6 AF48A03B BFD25E8C D0364141}, \end{aligned}$$

and the cofactor is $h = 1$. Thus, $E(\mathbb{F}_p)$ is isomorphic to the cyclic group $\mathbb{Z}/n\mathbb{Z}$.

4.7. The Tickets Generator

The ECC provides one of the main tools to perform computations in order to find the winner tickets. The set $\{1, \ldots, N\}$ of possible tickets yields the modulo N to be used in the extraction (cf. Section 4.8): it is chosen by the system and changes in time depending on the *inflation* level. The algorithm, moreover, is executed simultaneously, but independently, by all nodes that are provided with a utility, and the number of tickets that a node can extract depends on it: the greater the utility, the greater the number of tickets.

In particular, a ticket represents the number of operations (additions) to perform on $E(\mathbb{F}_p)$ starting from two given points $P(x_1, y_1)$ and $Q(x_2, y_2)$ on the curve: in this view, the ticket $m \in \{1, \ldots, N\}$ will correspond to the point $Q + mP \in E(\mathbb{F}_p)$, which expresses a specific token. The last ticket N is, hence, associated with the point $Q + NP \in E(\mathbb{F}_p)$. Note that, with n being prime, the coset $Q + \langle P \rangle$ always coincides with the whole group $E(\mathbb{F}_p)$ because P is different from the identity. Moreover, the nature of E as ECC guarantees that the points (tokens) in the plane are well-separated.

4.8. The Leader Selection Algorithm

Now, we describe, through an explicit example, some details of the core algorithm to select the leader. We remark that it is a “deterministic random algorithm” (i.e., same inputs, same outputs) able to give a highly uniform distribution of the outputs as if they were purely random actions.

The algorithm consists of the following steps:

1. Take the hash of the last block appearing in the blockchain, for example,

   4ceb86317d0d4dac6853663589ef02cc
   b67134cee75bb886a4410b7aedd0e109

2. Split the previous hash into eight parts:
   (a) 4ceb8631
   (b) 7d0d4dac
   (c) 68536635
   (d) 89ef02cc
   (e) b67134ce
   (f) e75bb886
   (g) a4410b7a
   (h) edd0e109
3. Convert each part into a binary code:
   - $A_1 = 01001100111010111000011000110001$
   - $A_2 = 01111101000011010100110110101100$
   - $A_3 = 01101000010100110110011000110101$
   - $A_4 = 10001001111011110000001011001100$
   - $A_5 = 10110110011100010011010011001110$
   - $A_6 = 11100111010110111011100010000110$
   - $A_7 = 10100100010000010000101101111010$
   - $A_8 = 11101101110100001110000100001001$
4. Compute the exclusive disjunction (XOR) of $A_i,\ i = 1, 2, \ldots, 8$, obtaining

$$A_1 \oplus A_2 \oplus \ldots \oplus A_8 = 11001000111000011100100101011111 \tag{9}$$

   The order in grouping operations does not affect the result: only the parity of the number of 1s present on each bit is decisive. We recall in fact that (B_n, XOR) and $(\mathbb{Z}/2\mathbb{Z})^n$ are canonically isomorphic abelian groups, where B_n is the n bit set.
5. Convert the binary code (9) into a decimal number, obtaining

$$3370240351 \tag{10}$$

6. Compute the modulo N value of (10). For example, if $N = 1000$, we obtain 351: this means that the ticket extracted is number 351 and the winner token is given by the point $Q + 351P$ on the ECC. Therefore, the address that owns the token thus identified will be the elected leader.

4.9. The Board Members

Many different methods can be used to determine board members. For example, it is possible to iteratively use the algorithm described in Section 4.8, or, more simply, it is possible to consider b tickets subsequent to the one extracted in point 6 above—that is, $352, 353, 354, \ldots, 351 + b$ in our example, where b is the number of board members. The sum operation on the elliptic curve prevents these tickets from matching consecutive tokens.

We now assume that the number b of the committee members must, in our example, belong to the discrete interval $[4 \ldots 1000]$. In fact, in the beginning, the system needs at least 4 nodes to be started. Subsequently, the number b of committee members increases as the number of nodes n increases according to a formula of the type

$$b = \gamma \cdot \sqrt{n},$$

where γ is a parameter that varies over time and serves to keep the number of members of the committee smaller or equal to 1000.

As for the number t_i of tickets that each lottery of each single node i distributes, it follows an exponential function of the type

$$t_i = \delta \cdot \exp(n)$$

where n is the current number of nodes and δ is a control parameter equal to 1 at the beginning, but then decreases to contain the creation of new tokens.

5. Randomness Ticket Extractions Simulations

Simulations were conducted to determine if the Board Election process is a Pseudo-random Generator (PRG). Specifically, 10 simulations were created, each consisting of the following: 10 different experiments (lottery extractions) were conducted, and the same number of tickets were retrieved from each, with the proportion altering for each simulation. In particular, only 10% of the 1000 tickets for each experiment were utilized in the first simulation; 20% for each experiment in the second simulation; 30% for each experiment in the third simulation; and so on, until reaching 100% in the final simulation. Consequently, there are a total of 10 simulations, each with 10 experiments, for a total of 100 experiments. Figures 1 and 2 depict the simulation results for 100 and 1000 produced tickets.

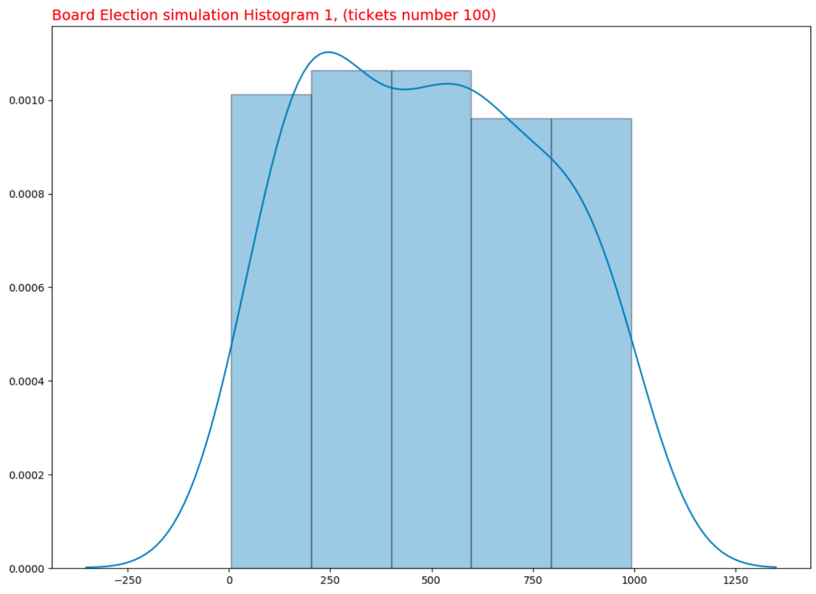

Figure 1. Ticket Distribution. 100 Tickets.

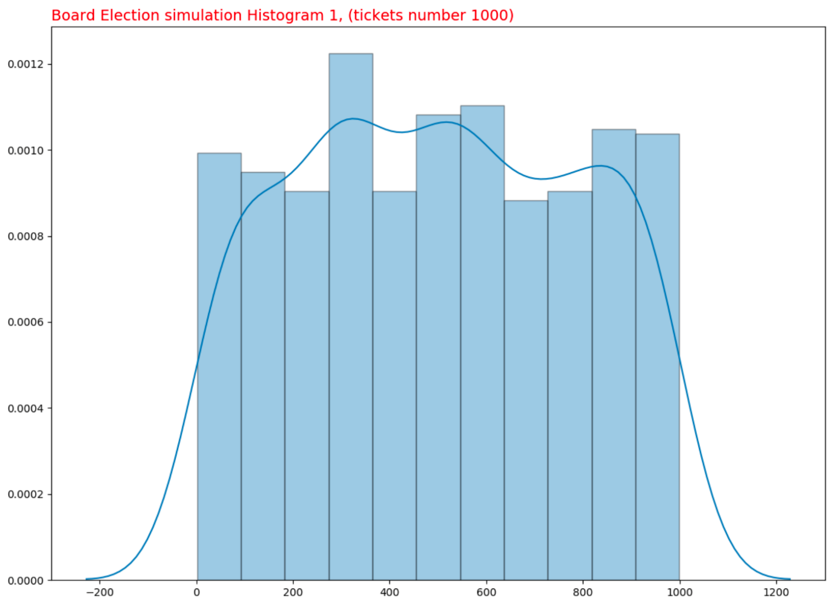

Figure 2. Ticket Distribution. 1000 Tickets.

All data from these simulations have been collected in order to construct statistical graphs. To better comprehend the underlying probability distribution of the data, a Density curve plot rather than a histogram has been used. The purpose of a density curve is to depict the underlying probability distribution of the data by sketching a continuous curve. This curve must be calculated from the data, and one of the most used methods—kernel density estimation—has been employed to accomplish this.

In kernel density estimation, a continuous curve (the kernel) with a small width (controlled by a parameter called *bandwidth*) is drawn at the position of each data point; then, the sum of all these curves is used to estimate the density. Typically, density curves are scaled so that the area under the curve equals 1. This indicates that there is no probability function on the Y axis for kernel density estimation, but rather, a density probability function. Regarding the bins to be utilized in the density plot graphs, the Freedman–Diaconis rule has been selected (FD rule). The FD rule is intended to minimize the discrepancy between the empirical and theoretical probability distributions' areas. The width of each bin is estimated as follows:

$$Bin_{width} = 2\frac{IQR(x)}{\sqrt[3]{n}} \tag{11}$$

where $IQR(x)$ is the interquartile range of the data and n is the number of observations in the sample x.

In addition, the Empirical Cumulative Density Function (ECDF) has been derived. A Cumulative Density Function is simply realized empirically. The term empirical refers to the fact that the function is constructed using an empirical sample measurement. The empirical distribution function is an approximation of the cumulative distribution function that produces the sample points. The definition is as follows: if $X_1, \dots, X_n$ are independent identically distributed (i.i.d.) real random variables, the *empirical distribution function* or *empirical cumulative distribution function (ECDF)* is a function $\widehat{F}_n$ with range $[0,1]$ given by

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{X_i \leq x}, \tag{12}$$

where $\mathbf{1}_{X_i \leq x}$ is the indicator function of the event $\{X_i \leq x\}$. Moreover, if we fix x, $\mathbf{1}_{X_i \leq x}$ is also a Bernoulli random variable with a parameter equal to the cumulative distribution function $F(x)$. Hence, $\widehat{F}_n(x)$ can be seen as an unbiased estimator for $F(x)$. For more details on basic probability and statistics, the reader can see [47].

Figures 3 and 4 depict the simulation results for 100 and 1000 extracted tickets.

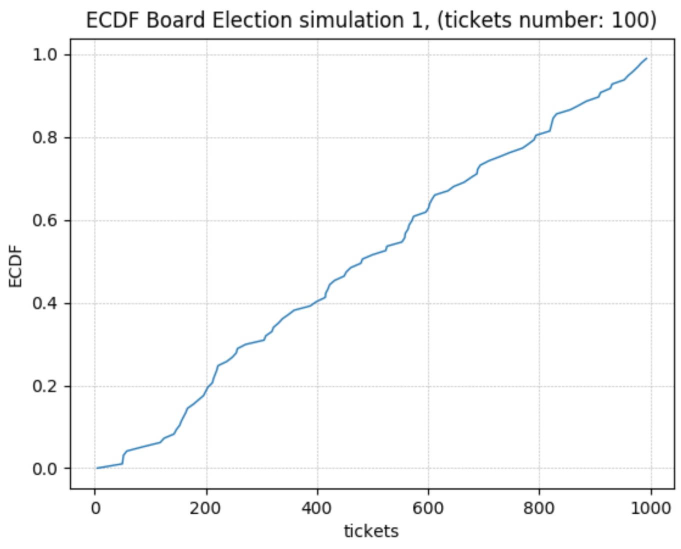

Figure 3. ECDF—Ticket Extraction. 100 Tickets.

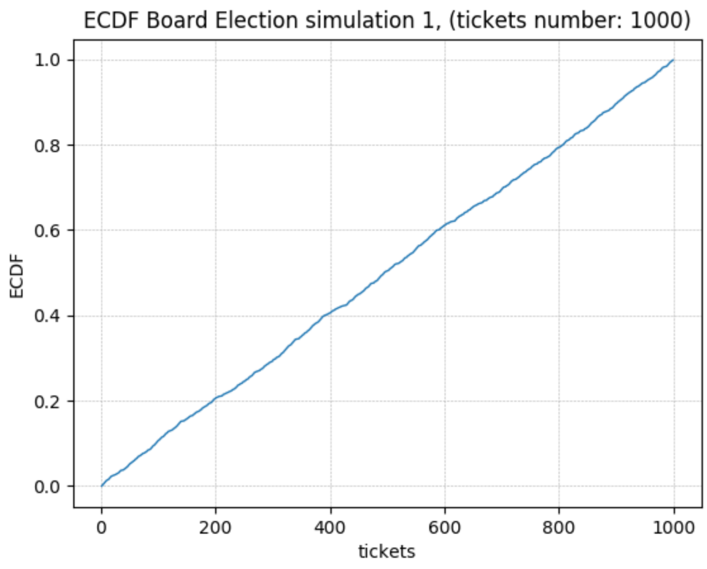

Figure 4. ECDF—Ticket Extraction. 1000 Tickets.

As predicted of a PRG, the algorithm outputs appear to follow a uniform distribution.

Analysis and Comparison

In this part, we conduct a comparative analysis of the various consensus methods that have been presented throughout this work. Table 2 provides a comprehensive comparison of the properties and performance of the consensus methods covered before.

Table 2. Consensus algorithms in terms of characteristics and performance.

	PoW	**PoS**	**PoA**	**NFP**
Type of Blockchain	Permissionless	Permissionless and Permissioned	Permissionless and Permissioned	Permissionless and Permissioned
Electing miners based on	Solving Difficulty Hash	Stake Owned	Solving Difficulty Hash	Stake Owned and Random Lottery
Model of Trust	Untrusted	Untrusted	Not Available	Self-Verifiable
Transaction Finality	Probabilistic	Probabilistic	Not Available	Immediate
Decentralization Structure	Strong	Strong	Strong	Strong
Properties of Distributed Consensus	Probabilistic	Probabilistic	Probabilistic	Deterministic
Reward	Yes	Yes	Yes	Yes
Control of acceptance of the adversary	Less than 25% computing power	Less than 51% stack	50% online stake	Less than 33.3% nodes
Fees on Transactions	Yes, for all miners	Yes, for all miners	Yes, for all miners and lucky stakeholders	Yes, for all elected board
Speed of tx verification (per s.)	Greater than 100 s	Less than 100 s	Not Available	Less than 10 s.
Throughput (tx/s)	Less than 100	Less than 1000	Not Available	Less than 2000
Speed Block creation	Low	High	High	High
Consumption of Energy	High	Sometimes less than PoW	Low	Low
Scalability	Moderate	Strong	Strong	Strong
Crash fault tolerance	50%	50%	50%	33%
Byzantine fault tolerance	50%	50%	50%	33%

In blockchain, consensus algorithms are utilized as a vital technology, and their algorithms guarantee the technology's steady functioning. Through the consensus protocol, consensus algorithm nodes agree on a specific value or transaction. However, research in this field is currently ongoing. We discussed various common blockchain consensus algorithms as well as a comprehensive comparison and analysis of their properties in relation to our Neural Fairness Protocol. After analysis, we have determined that the Neural Fairness Protocol offers a high throughput, but it must be evaluated in a production context in order to evaluate its performance in further detail.

6. Rewards

Nowadays, there are different opinions and philosophical discussions about rewards and incentives, and how to keep public blockchains alive.

We revert to thought leader and Bitcoin creator Satoshi Nakamoto, who expressed that an ecosystem of nodes could survive only if an incentive is provided to those who contribute to the network.

Agreeing with this opinion, the Neural Fairness Protocol rewards nodes that participate in the committee when a block is elected and written to the blockchain. Based on the fairness activities (score) that each node has reached in contributing to a block election, the collected fees will be shared across all the committee. Therefore, once the block is written into the blockchain, the nodes that were part of the committee will receive in their own wallet the fair calculated amount of tokens.

7. Conclusions and Future Work

Various works have effectively targeted the confidentiality of Bitcoin addresses by examining blockchain transactions [48–52]. By merging the benefits of ECDSA [35] and cryptography into a single system, we have proposed in this study the Neural Fairness Protocol, an innovative consensus process that greatly outperforms prior work. We provide a novel strategy for the next forger election that improves robustness, anonymity, and denial of responsibility. By evaluating the existing traditional blockchain, we demonstrate how our single-block production method ensures anonymity. The threshold ECDSA algorithm created as part of the prototype is efficient and safe against malicious attackers, and may be utilized outside the scope of our research, for instance, to protect Bitcoin wallets. Our work is immediately compatible with other cryptocurrencies that employ the same ECDSA primitive, such as Litecoin and Mastercoin, despite the fact that we concentrated on traditional blockchain transactions such as the Bitcoin blockchain.

As the research and development of the Neural Fairness Protocol progresses, we must conduct a thorough review of our implemented prototype to ensure the protocol's confidentiality, security, and efficacy. We are attempting to leverage and integrate technology such as neural networks and cryptography to create a fully fair protocol for the subsequent blockchain type.

Author Contributions: Conceptualization, F.C., G.d. and E.Z.; Funding acquisition, G.d.; Investigation, F.C. and E.Z.; Methodology, F.C. and E.Z.; Project administration, G.d.; Software, E.Z.; Supervision, G.d.; Writing—original draft, F.C. and E.Z.; Writing—review & editing, F.C. and E.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported with the sponsorship of BEEZ s.r.l., Cosenza, Italy.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three-letter acronym
LD	Linear dichroism
PoS	Proof of Stake
PoA	Proof of Autority
PoAC	Proof of Accuracy
PoC	Proof of Capacity
VCL	verifiable cryptographic lottery
ECC	elliptic curve cryptography
SUHA	Simple Uniform Hashing Assumption
NFP	Neural Fairness Protocol
ECDH	Elliptic-curve-based Diffie-Hellman Key Agreement
CL	cryptographic lottery
PoI	Proof of Interest
SOM	Self-Organizing-Map
BUM	Best Unit Match
TUSs	targets utility stakeholders
MDP	Markov Decision Process
DQN	Deep-Q-Network
RF	Reinforcement Learning
HP	Honest Player
BP	Bad Player
CoI	Conflict of Interest
CG	Conflict Graph
MWIS	Maximum Weight Independent Set
PRG	Pseudorandom Generator
FD	Freedman–Diaconis
ECDF	Empirical Cumulative Density Function

References

1. Chaum, D. Blind Signatures for Untraceable Payments. *Adv. Cryptol. Proc. Crypto* **1982**, *82*, 199–203. [CrossRef]
2. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 6 October 2020).
3. Lamport, L.; Shostak, R.; Pease, M. The Byzantine general problem. *ACM Trans. Programm. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]
4. Szabo, N. Bit Gold. 2005. Available online: https://nakamotoinstitute.org/bit-gold/ (accessed on 5 June 2021).
5. Vishnumurthy, V.; Chandrakumar, S.; Sirer, E. KARMA : A Secure Economic Framework for Peer-To-Peer Resource Sharing. In Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, USA, 5–6 June 2003. Available online: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html. (accessed on 13 June 2021).
6. Wei, D. B-Money. 1998. Available online: https://nakamotoinstitute.org/b-money/ (accessed on 10 July 2021).
7. Dwork, C.; Goldberg, A.; Naor, M. On Memory-Bound Functions for Fighting Spam. In *Advances in Criptology—CRYPTO 2003*; Lecture Notes in Computer Science; Boneh, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2729; pp. 380–409. [CrossRef]
8. Back, A. Hashcash—A Denial of Service Counter-Measure. Technical Report, Available online: www.hashcash.org/hashcash.pdf (accessed on 15 April 2021).
9. Begicheva, A.; Kofman, A. Fair Proof of Stake. In *Fair Block Delay Distribution, in Proof-of-Stake Project*; Waves Platform: Moscow, Russia, 2018. [CrossRef]
10. Joshi, S. Feasibility of Proof of Authority as a Consensus Protocol Model. *arXiv* **2021**, arXiv:2109.02480v1.
11. Aponte-Novoa, F.; Villanueva-Polanco, R. On Proof-of-Accuracy Consensus Protocols. *Mathematics* **2022**, *10*, 2504. [CrossRef]
12. Dziembowski, S.; Faust, S.; Kolmogorov, V.; Pietrzak, K. Proofs of Space. In *Advances in Cryptology—CRYPTO 2015*; Lecture Notes in Computer Science; Gennaro, R.; Robshaw, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9216; pp. 585–605. [CrossRef]
13. Bashar, G.; Hill, G.; Singha, S.; Marella, P.; Dagher, G.G.; Xiao, J. Contextualizing Consensus Protocols in Blockchain: A Short Survey. In Proceedings of the 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), Los Angeles, CA, USA, 12–14 December 2019; pp. 190–195. [CrossRef]

14. Shrimali, B.; Patel, H.B. Blockchain state-of-the-art: Architecture, use cases, consensus, challenges and opportunities. *J. King Saud-Univ.-Comput. Inf. Sci.* 2021, *in press*. [CrossRef]
15. Ferdous, M.S.; Chowdhury, M.J.M.; Hoque, M.A. A survey of consensus algorithms in public blockchain systems for crypto-currencies. *J. Netw. Comput. Appl.* **2021**, *182*, 103035. [CrossRef]
16. Xiong, H.; Chen, M.; Wu, C.; Zhao, Y.; Yi, W. Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms. *Future Internet* **2022**, *14*, 47. [CrossRef]
17. Oyinloye, D.P.; Teh, J.S.; Jamil, N.; Alawida, M. Blockchain Consensus: An Overview of Alternative Protocols. *Symmetry* **2021**, *1363*, 13. [CrossRef]
18. Sriman, B.; Ganesh Kumar, S.; Shamili, P. Blockchain Technology: Consensus Protocol Proof of Work and Proof of Stake. In *Intelligent Computing and Applications*; Advances in Intelligent Systems and Computing; Dash, S.S.; Das, S.; Panigrahi, B.K., Eds.; Springer: Singapore, 2021; Volume 1172. [CrossRef]
19. Skh Saad, S.M.; Radzi, R. Comparative Review of the Blockchain Consensus Algorithm Between Proof of Stake (POS) and Delegated Proof of Stake (DPOS). *Int. J. Innov. Comput.* **2020**, *10*. [CrossRef]
20. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd revised and extended ed.; MIT Press: Cambridge, MA, USA, 2009.
21. Ferguson, N.; Schneier, B.; Kohno, T. *Cryptography Engineering: Design Principles and Practical Applications*; John Wiley & Sons: Indianapolis, IN, USA, 2010. [CrossRef]
22. Lewis, H.; Zax, R. *Essential Discrete Mathematics for Computer Science*; Princeton University Press: Princeton, NJ, USA, 2019.
23. Brown, D.R.L. Standards for Efficient Cryptography 2 (SEC 2): Recommended Elliptic Curve Domain Parameters, ver. 2.0, Certicom Research, 2010. Available online: https://www.secg.org/sec2-v2.pdf (accessed on 25 March 2021).
24. Caldarola, F.; D'Atri, G.; Maiolo, M.; Pirillo, G. New algebraic and geometric constructs arising from Fibonacci numbers. In honor of Masami Ito. *Soft Comput.* **2020**, *24*, 17497–17508. [CrossRef]
25. Caldarola, F.; d'Atri, G.; Maiolo, M.; Pirillo, G. The sequence of Carboncettus octagons. In Proceedings of the 3rd International Conference "Numerical Computations: Theory and Algorithms", Crotone, Italy, 15–21 June 2019; Sergeyev, Y.D.; Kvasov, D., Eds.; Springer: Cham, Switzerland, 2020; Volume 11973; LNCS; pp. 373–380. [CrossRef]
26. Coleman, D.A.; Dugan, C.J.; McEwen, R.A.; Reiter, C.A.; Tang, T.T. Periods of (q, r)-Fibonacci sequences and elliptic curves. *Fibonacci Q.* **2006**, *44*, 59–70.
27. Karuna Kamath, K.; Shankar, B.R. Elliptic curves and Fibonacci sequences. *Int. Math. Comput.* **2009**, *4*, 20–25.
28. Liu, S.; Qi, G.; Wang, X.A. Fast elliptic curve algorithm using deformed Fibonacci-type series. *Int. J. Embed. Syst.* **2018**, *10*, 104–112. [CrossRef]
29. Caldarola, F.; d'Atri, G.; Mercuri, P.; Talamanca, V. On the Arithmetic of Knuth's Powers and Some Computational Results About Their Density. In Proceedings of the 3rd International Conference "Numerical Computations: Theory and Algorithms", Crotone, Italy, 15–21 June 2019; Sergeyev, Y.D.; Kvasov, D., Eds.; Springer: Cham, Switzerland, 2020; Volume 11973; LNCS; pp. 381–388. [CrossRef]
30. Leonardis, A.; d'Atri, G.; Caldarola, F. Beyond Knuth's notation for unimaginable numbers within computational number theory. *Int. Electron. J. Algebra* **2022**, *31*, 55–73. [CrossRef]
31. Caldarola, F.; d'Atri, G.; Pellegrini, M. Combinatorics on n-sets: Arithmetic Properties and Numerical Results. In Proceedings of the 3rd International Conference "Numerical Computations: Theory and Algorithms", Crotone, Italy, 15–21 June 2019; ; Sergeyev, Y.D.; Kvasov, D., Eds.; Springer: Cham, Swizterland, 2020; Volume 11973; LNCS; pp. 389–401. [CrossRef]
32. Feng, Q.; He, D.; Zeadally, S.; Khan, M.K.; Kumar, N. A survey on privacy protection in blockchain system. *J. Netw. Comput. Appl.* **2019**, *126*, 45–58. [CrossRef]
33. Lashkari, B.; Musilek, P. A comprehensive review of blockchain consensus mechanisms. *IEEE Access* **2021**, *9*, 43620–43652. [CrossRef]
34. Pass, R.; Shi, E. The Sleepy Model of Consensus. *Advances in Criptology—ASIACRYPT 2017*; Takagi, T.; Peyrin, T., Eds.; Springer: Cham, Switzerland, 2017; Volume 10625; LNCS; pp. 380–409. [CrossRef]
35. Menezes, A. The Elliptic Curve Discrete Logarithm Problem: State of the Art. *Advances in Information and Computer Security—IWSEC 2008*; Lecture Notes in Computer Science; Matsuura, K.; Fujisaki, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5312; p. 218. [CrossRef]
36. Lovecruft, I.; de Valence, H. The Ristretto Group. 2008. Available online: https://ristretto.group/ristretto.html (accessed on 15 April 2021).
37. Grassi, L.; Kales, D.; Khovratovich, D.; Rechberger, C.; Roy, A.; Schofnegger, M. Starkad and Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. *IACR Cryptol. ePrint Arch.* **2019**, 458.
38. Bünz, B.; Bootle, J.; Boneh, D.; Poelstra, A.; Wuille, P.; Maxwell, G. Bulletproofs: Short Proofs for Confidential Transactions and More. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018; pp. 315–334. [CrossRef]
39. Bellini, E.; Murru, N. A Multi-factor RSA-like Scheme with Fast Decryption Based on Rédei Rational Functions over the Pell Hyperbola. In Proceedings of the 3rd International Conference "Numerical Computations: Theory and Algorithms", Crotone, Italy, 15–21 June 2019; Sergeyev, Y.D.; Kvasov, D., Eds.; Springer: Cham, Switzerland, 2020; Volume 11973; LNCS; pp. 343–357. [CrossRef]

40. Langville, A.; Meyer, C.; Fernández, P. Google's Pagerank and Beyond: The Science of Search Engine Rankings. *Math. Intell.* **2008**, *30*, 68–69. [CrossRef]
41. Sanghi, N., Markov Decision Processes. In *Deep Reinforcement Learning with Python*; Apress: Berkeley, CA, USA, 2021; pp. 19–48. [CrossRef]
42. Keil, J.; Mitchell, J.; Pradhan, D.; Vatshelle, M. An algorithm for the maximum weight independent set problem on outerstring graphs. *Comput. Geom.* **2017**, *60*, 19–25. [CrossRef]
43. Mari, M. *Study of Greedy Algorithm for Solving Maximum Independent Set Problem*; Technical Report; ENS Rennes, University of Liverpool: Liverpool, UK, 2017.
44. Chaudhry, A.W. *A New Algorithm for Solving the Maximum Independent Set Problem*; Griffith University: Gold Coast, Australia, 2019.
45. Goldberg, M.; Spencer, T. An Efficient Parallel Algorithm That Finds Independent Sets Of Guaranteed Size. *SIAM J. Discrete Math.* **1993**, *6*, 443–459. [CrossRef]
46. Hifi, M.; Otmani, N. An algorithm for the disjunctively constrained knapsack problem. *Int. J. Oper. Res.* **2012**, *13*, 22–43. [CrossRef]
47. van der Vaart, A.W. *Asymptotic Statistics*; Cambridge University Press: Cambridge, UK, 1998.
48. Androulaki, E.; Karame, G.O.; Roeschlin, M.; Scherer, T.; Capkun, S. Evaluating User Privacy in Bitcoin. In *Financial Cryptography and Data Security. FC 2013*; Lecture Notes in Computer Science; Sadeghi, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7859. [CrossRef]
49. Ron, D.; Shamir, A. Quantitative Analysis of the Full Bitcoin Transaction Graph. In *Financial Cryptography and Data Security*; FC 2013; Lecture Notes in Computer Science; Sadeghi, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7859. [CrossRef]
50. Spagnuolo, M.; Maggi, F.; Zanero, S. BitIodine: Extracting Intelligence from the Bitcoin Network. *Financial Cryptography and Data Security. FC 2014*; Christin, N.; Safavi-Naini, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8437, Lecture Notes in Computer Science, pp. 457–468. [CrossRef]
51. Zanardo, E. Bitnocolo-Anti Money Laundering (AML) Tool for Blockchain Transactions. 2020.
52. Adu-Gyamfi, D.; Kwansah Ansah, A.K.; Armah, G.K.; Alornyo, S.; Adom, D.K.; Zhang, F. Towards bitcoin transaction anonymity with recurrent attack prevention. *Int. J. Syst. Assur. Eng. Manag.* **2022**, *13*, 1–17. [CrossRef]