



## Article

# On Proof-of-Accuracy Consensus Protocols

Fredy Andres Aponte-Novoa <sup>1,2,†</sup>  and Ricardo Villanueva-Polanco <sup>1,\*,†</sup> 

<sup>1</sup> Department of Computer Science and Engineering, Universidad del Norte, Barranquilla 081007, Colombia; faponte@uninorte.edu.co

<sup>2</sup> Department of Systems Engineering, Universidad Santo Tomás, Tunja 150003, Colombia

\* Correspondence: rpolanco@uninorte.edu.co

† These authors contributed equally to this work.

**Abstract:** Consensus protocols are a fundamental part of any blockchain; although several protocols have been in operation for several years, they still have drawbacks. For instance, some may be susceptible to a 51% attack, also known as a majority attack, which may suppose a high risk to the trustworthiness of the blockchains. Although this attack is theoretically possible, executing it in practice is often regarded as arduous because of the premise that, with sufficiently active members, it is not ‘straightforward’ to have much computing power. Since it represents a possible vulnerability, the community has made efforts to solve this and other blockchain problems, which has resulted in the birth of alternative consensus protocols, e.g., the proof of accuracy protocol. This paper presents a detailed proposal of a proof-of-accuracy protocol. It aims to democratize the miners’ participation within a blockchain, control the miners’ computing power, and mitigate the majority attacks.

**Keywords:** alternative consensus protocols; blockchain; consensus protocol; proof of accuracy

**MSC:** 94A60; 68M14; 14G50



**Citation:** Aponte-Novoa, F.A.; Villanueva-Polanco, R. On Proof-of-Accuracy Consensus Protocols. *Mathematics* **2022**, *10*, 2504. <https://doi.org/10.3390/math10142504>

Academic Editors: Jan Lansky and Antanas Cenys

Received: 16 June 2022

Accepted: 14 July 2022

Published: 19 July 2022

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Blockchain technology is considered one of the most prominent post-internet disruptive computing paradigms [1,2], featuring unique attributes that make it an ideal set of techniques to register, verify, and manage transactions [3]. A blockchain network performs the secure administration of the shared ledger, where transactions are verified and stored in a network without a central authority [4]. Applications that use blockchain, such as cryptocurrencies, decentralized finance applications, video games, and many others, trust that blockchain will prevent problems such as fraud, thanks to the integrated cryptographic mechanisms provided by the data structure and the consensus protocol [5].

Since blockchain is an emerging technology, its applicability to new domains and challenges is constantly growing. The paper [6] presented a taxonomy of blockchain-based applications and an analysis of blockchain challenges regarding security and performance. Its study covered 96 papers categorized into 7 application domains: finance (e.g., [7]), achievement records (e.g., [8]), energy (e.g., [9]), healthcare (e.g., [10]), manufacturing (e.g., [11]), supply chain (e.g., [11]), shipping and delivery (e.g., [12]), and sustainability (e.g., [13]). Regarding security challenges, they highlighted majority attacks [5], DDoS attacks, selfish mining, and others [6]. Concerning performance challenges, they focused on throughput and latency in some blockchains, as well as resource and energy management issues [6].

The 51% attack (also called majority attack) is categorized into hash-based vulnerabilities (“hash-based attack”). It entails one or more miners taking control of at least 51% of the mined hash or computation in the blockchain network [14]. Due to this vulnerability, an attacker can perform attacks that involve canceling transactions, creating forks in the blockchain, selfish mining, and double-spending virtual currency [15].

Consensus protocols are at the core of the inner-working of any blockchain. Proof-of-work (PoW) and proof-of-stake (PoS) protocols are the most popular; however, they

still have drawbacks. In particular, the consensus mechanism PoW is inefficient regarding energy consumed by its participants [16]. As a result of the efforts to improve the PoW and PoS protocols, the so-called alternative consensus protocols were born [17].

As evinced in Section 2, there is little research on proof-of-accuracy consensus protocols. In particular, its study and development have been theoretical and are part of the so-called blockchain consensus alternative protocols. According to [18], realizing it requires some components, such as selection of a coordinator, generation of a secret, generation and distribution in the network of parts of the secret, and competition between the participants to find the parts and reconstruct the secret. However, no proposal has presented a concrete protocol so far. Hence, this paper aims to present a detailed proposal for the formalization of what is called a proof-of-accuracy protocol.

The main contribution of this paper is to introduce a proof-of-accuracy protocol. We present our proposed protocol progressively, starting with an initial blueprint (based on different components described in [18] and its drawbacks), which is improved further concerning security. Earlier versions of our protocol feature the following phases: selection of a coordinator, generation of a secret, generation and distribution of parts of the shared secret in the network, and competition between the participants to find the shares to reconstruct the secret (proof of work component). However, our last version (see Section 3.7) removes the need for a coordinator and combines the proof-of-work feature with access to random locations to improve the protocol's resistance to majority attacks.

The remainder of the article is structured as follows: Section 2 presents an overview of consensus algorithms. Section 3 presents our protocol proposal, introducing it progressively from earlier versions. This section presents the notation we use to describe our protocol and its earlier designs, a general description of a proof-accuracy protocol, our assumptions, and the description of our proposed protocol (and its earlier versions). Section 4 presents an analysis of our proposed protocol. In particular, we present a deep analysis of its mining process, computational costs, and security. Section 5 presents a proof-of-concept implementation of our proposed protocol. Section 6 presents a qualitative comparison between our protocol and other consensus protocols proposed in the literature. In Section 7, we draw our conclusions and describe future research directions.

## 2. Consensus Protocols

### 2.1. Original Proof of Work

The nodes must agree when/if a node may add a new block to a blockchain. The PoW consensus algorithm requires the nodes to solve a puzzle, then the first node to solve the puzzle wins the right to add the new block to the chain. This node is named miner, and the effort of solving the puzzle is named mining. Furthermore, the puzzle's difficulty may be modified each time that a fixed number of blocks is added to the chain [19].

### 2.2. PoS-Based

The algorithms based on proof of stake (PoS) are based on the idea of betting to determine the node that wins the right to mine the next block in the chain. Employing participation (as proof) is favorable: whatever node that has previous participation is more trustworthy. Therefore, it is expected that a node having a large portion of its gains on the chain will not execute some dishonest move to attack the same chain. Furthermore, the use of PoS indicates that there has to be at least 51% of all betting in the network to execute a double-spend attack, which is very difficult [19].

### 2.3. Hybrid form of PoW and PoS

"Coin age" is a new concept [20]; it is calculated for each miner as a product of his/her stake by the time that the miner owns it. The node that has the right to mine a new block in the chain must create a special block named a coin stake; it contains multiple transactions and a special transaction from that miner to itself. The quantity of money expended on the transaction offers the miner additional possibilities to solve the puzzle and add a new

block, as in PoW. If a miner spends more money on the transaction, the miner has more of a chance to solve the puzzle. If a node solves a puzzle, it obtains 1% of the coins that they have spent in the transaction, and the accumulated coin age by these coins is reset to zero [19].

Another different proposal [21] does not utilize the coinage because it is supposed that, using the coinage, the attacker can be given the opportunity to collect sufficient value to fool the network. Furthermore, another possible problem occurs when some miners save their stakes until they have a considerable number of coins. During this period they stay outside of the verification system. Thus, the proposal by Vasin [21] consists in utilizing pure participation instead of the coin age to offer the miners the possibility of adding a new block.

#### 2.4. Other Kinds of Proof-Based Consensus Algorithms

The excessive energy demand that needs PoW protocols to find the nonce value is possibly their main drawback. Moreover, such a calculation may not offer any benefit to the user as pinpointed by Blocki and Zhou in [22] and by King in [23]. To solve this problem, Blocki and Zhou [22] suggested using some types of puzzles for education and social activities; these puzzles are challenging for people but easy to solve for computers. This proposal is more for miners because not all have modern hardware [19].

Various authors have presented other evidence-based consensus protocols that do not utilize the ideas of PoW and PoS. Examples of these protocols are proof of burn [24], proof of space [25], proof-of-QoS (PoQ) [26], and a fair selection protocol [27]. These and other consensus protocols are presented in [16].

#### 2.5. Alternative Protocols

Alternative consensus protocols have originated as responses to different efforts to improve the traditional consensus protocols. The works carried out on these alternative protocols are little known. We can evidence this by the number of recent scientific publications. For example, when consulting the number of publications related to alternative consensus protocols and blockchain in Scopus, IEEE Xplore, and Web of Science, we found only two publications, both made in the year 2021.

In [17], they made a general description of alternative consensus protocols proposed between 2019 and 2021. They classified the alternative consensus protocols according to how the winner node was selected as follows: consensus protocol based on effort or work (CPE), consensus protocol based on wealth or resources (CPW), consensus protocol based on past behavior or reputation (CPPB), and consensus protocol based on representation (CPR). Furthermore, the authors of [28] presented a modular version of PoS-based blockchain systems called e-PoS, which resisted the centralization of network resources by expanding mining opportunities to a more extensive set of participants. In addition to the few publications on the subject, the interest in studying these protocols lies in their design features, which can guide future research.

#### 2.6. Proof of Accuracy Protocol

The authors of [18] presented novel ideas for creating new consensus protocols, introducing two possible protocols: protocol simple tickets and proof of accuracy consensus protocol (PoAc); these ideas are theoretical with no concrete implementation.

PoAc features an effort or work (CPE) component since the selection of the node that wins the right to add the new block to the network is not only based on calculating the solution of a problem with a certain threshold of computational complexity but also should include a proof that the winning node has the necessary data pieces for the calculation of the solution of the problem.

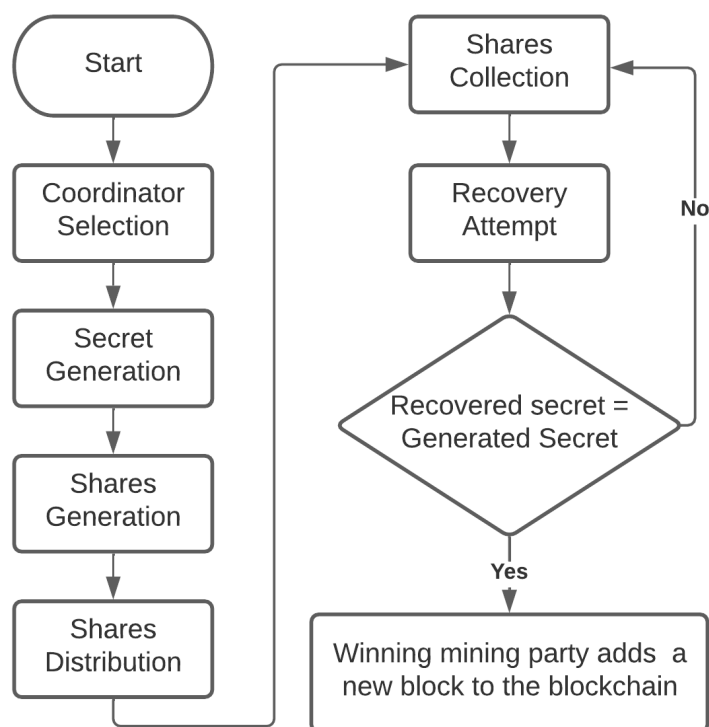
These data pieces must be correctly defined and follow a random distribution to ensure that the task of collecting these pieces is stochastic and feasible in a given interval of

time, which may be increased by adding some decoy pieces and distributing them with the genuine data pieces on the network [17].

The network participants vie for accessing the data pieces needed to solve the mining problem, which helps participants with little computing skills have the opportunity to win the proof-of-work and add a new block to the chain. Figure 1 shows the flow of the proof of accuracy consensus protocol, which features a random selection of a coordinator, who generates a secret, divides it into shares, and distributes them among participants; the mining process then starts and consists of accessing these shareholders to reconstruct the secret. The mining party reconstructing the secret will acquire the right to add a new block to the blockchain.

A recent paper [29] proposed a delegated proof of accessibility (DPoAC) protocol, mostly based on the previous idea. It employed secret sharing, PoS with random selection, and an interplanetary file system (IPFS). This protocol is similar to our initial design presented in Section 3.5 and follows a similar flow as shown in Figure 1. The main difference is that the coordinator stores the  $n$  shares of the secret in different  $n$  nodes on the IPFS network. For a mining party to acquire block creation rights, the party has to access these shareholders to reconstruct the secret.

As analyzed in Section 3.5.1, this approach has a drawback in that it heavily relies on the coordinator, meaning this node gains too much knowledge of the secret and may take advantage of it to favor any mining party.



**Figure 1.** Proof of accuracy flowchart.

### 3. Proposed Protocol

In this section, we introduce our proposed protocol. In Section 3.1, we introduce the notation that we will use throughout the section. In Sections 3.5 and 3.6, we describe earlier versions of our proposed protocol, highlighting their weaknesses and disadvantages. These earlier versions serve as a base to introduce our proposed protocol. Finally, we present our proposed protocol in Section 3.7.

### 3.1. Notation

We introduce the notation that we will use throughout the section. Specifically, Table 1 summarizes our notation.

**Table 1.** Summary of notation.

Symbol	Description
$t \in \mathbb{N}$	A positive integer.
$n \in \mathbb{N}$	A positive integer.
$m \in \mathbb{N}$	A positive integer.
$m_0 \in \mathbb{N}$	A positive integer.
$m_1 \in \mathbb{N}$	A positive integer.
$\mathbb{G} \in \mathbb{N}$	Denotes a cyclic group of order $q$
$q \in \mathbb{N}$	A prime number
$p = 2q + 1$	A safe prime number
$g \in \mathbb{G}$	A public generator of $\mathbb{G}$
$\mathbb{Z}_p$	The ring of integers modulo $p$
$\mathbb{ID}_m = \{1, 2, \dots, m\}$	The set of $m$ identifiers
$i \in \mathbb{ID}_m$	An identifier in $\mathbb{ID}_m$
$s_i(j) : \mathbb{ID}_m \setminus \{i\} \rightarrow \{1, -1\} \subseteq \mathbb{Z}_q \text{ [30]} \quad s_i(j) = \begin{cases} 1 & \text{if } i > j \\ -1 & \text{if } i < j \end{cases} \quad (1)$	
$\mathcal{H}_0 : \{0, 1\}^* \rightarrow \{0, 1\}^{m_0}$	Denotes a cryptographic hash function
$\mathcal{H}_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{m_1}$	Denotes a cryptographic hash function
$0 \leq p_{\min} \leq 1 \in \mathbb{R}$	Denotes a probability (protocol parameter)

### 3.2. General Description

The progressive versions of our protocol are built upon the cryptographic components required to compose a proof-of-accuracy (PoAc) protocol described by [18]. In particular, according to [18], a proof-of-accuracy (PoAc) protocol features the selection of a coordinator among all the participants, the joint generation of a secret by all the participants, the generation of shares of the generated secret, decoy shares, the distribution of all of them over the network participants, the mining process among the mining parties to reconstruct the generated secret, and the proof of recovering the secret by the winning party. We next describe the cryptographic tools we used.

#### 3.2.1. Single Broadcast-Based Joint Random Number Generation Protocol

Our proposed protocol uses a joint random number generation protocol as described in [30,31]. According to their designers, these protocols do not require a secure network and need one transmission per network node. The protocol [30] features additive aggregation instead of a multiplicative aggregation as in the protocol presented in [31].

The arithmetic carried out by  $m$  participants in the protocol [30] works over a cyclic group  $\mathbb{G} \subseteq \mathbb{Z}_p$  generated by  $g$  with order  $q$ , where  $q$  and  $p = 2q + 1$  are prime numbers. Each participant generates a Diffie–Hellman-like key pair  $(\kappa_i, pk_i = g^{\kappa_i})$  and shares its public key with the other network participants. With the set of public keys, the participant  $i$  computes  $R_i = \sum_{j=1, j \neq i}^m s_i(j) pk_j^{\kappa_i} \in \mathbb{Z}_p$ , generates  $c_i \in \mathbb{Z}_p$ , and calculates  $\gamma_i = c_i + R_i$ . At the last step, a randomly chosen coordinator takes the role of the combiner and collects all the  $\gamma_i$  values, and computes  $\alpha = \sum_{i=1}^m \gamma_i = \sum_{i=1}^m c_i + \sum_{i=1}^m R_i = \sum_{i=1}^m c_i$ , since  $\sum_{i=1}^m R_i = 0$ , which will be the input secret passed to the next sub-protocol.

#### 3.2.2. Shamir's Secret Sharing Scheme

Shamir's secret sharing scheme aims to divide a secret  $\alpha$  in  $s$  parts  $(\alpha_1, \alpha_2 \dots \alpha_s)$  so that with any  $t$  of the  $s$  parts,  $\alpha$  can be reconstructed, but every set of  $t - 1$  reveals nothing about  $\alpha$  [32]. Shamir's secret sharing scheme stems from a general fact of polynomial interpolation; A polynomial of maximum degree  $t - 1$  defined over a field is fully determined by  $t$  points

of the polynomial. In our particular case, we work it over  $\mathbb{Z}_q$ , which is a field when  $q$  is a prime number [32].

We call the  $s$  parts the genuine parts, the valid ones to reconstruct the secret; Also,  $n - s$  non-genuine parts are generated and distributed among the participants. Combining genuine and non-genuine parts allows for adjusting the difficulty in collecting  $t$  genuine parts to recover the secret. Algorithms 1 and 2 describe the inner workings of Shamir's secret sharing scheme.

---

**Algorithm 1** generates the  $t$ -out-of- $s$  sharing of  $\alpha$

---

```

1: function  $G_{sh}(s, t, \alpha)$ 
2:   choose  $a_1, \dots, a_{t-1} \leftarrow \mathbb{Z}_q$  at random and define a polynomial
      
$$f(x) := a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \dots + a_1x + \alpha \in \mathbb{Z}_q.$$

3:   for  $i \leftarrow 1$  to  $s$  do
4:     select  $x_i$  randomly from  $\mathbb{Z}_q$ , such that  $x_i \neq x_j$  for all  $j \in \{1, \dots, i-1\}$ 
5:      $y_i \leftarrow f(x_i)$ 
6:      $\alpha_i \leftarrow (x_i, y_i)$ 
7:   end for
8:   return  $(\alpha_1, \alpha_2, \dots, \alpha_s)$ 
9: end function

```

---



---

**Algorithm 2** recovers  $\alpha$  given  $t$  genuine shares

---

```

1: function  $C_{sh}(\alpha_1 = (x_1, y_1), \dots, \alpha_t = (x_t, y_t))$ 
2:    $\alpha \leftarrow 0 \in \mathbb{Z}_q$ 
3:   for  $i \leftarrow 1$  to  $t$  do
4:      $\lambda_i \leftarrow 1 \in \mathbb{Z}_q$ 
5:     for  $j \leftarrow 1$  to  $t$  do
6:       if  $i \neq j$  then
7:          $\lambda_i \leftarrow \lambda_i \cdot \frac{-x_j}{x_i - x_j}$ 
8:       end if
9:     end for
10:     $\alpha \leftarrow \alpha + y_i \cdot \lambda_i$ 
11:  end for
12:  return  $\alpha$ 
13: end function

```

---

### 3.2.3. Schnorr Non-Interactive Zero-Knowledge Scheme

Schnorr non-interactive zero-knowledge (NIZK) scheme is a non-interactive variant of the three-pass Schnorr identification scheme. The Schnorr NIZK scheme allows a prover to prove to any verifier their knowledge of a discrete logarithm without leaking any information about its value [32]. Algorithm 3 shows how a proof gets generated, while Algorithm 4 shows how a proof gets verified.

---

**Algorithm 3** generates a proof

---

```

1: function  $GENPROOF(\alpha, u, id)$ 
2:    $\alpha_t \xleftarrow{R} \mathbb{Z}_q$ 
3:    $u_t \leftarrow g^{\alpha_t}$ 
4:    $c \leftarrow H_0(g, u_t, u, id)$ 
5:    $\alpha_z \leftarrow \alpha_t + c \cdot \alpha$ 
6:   return  $(id, u_t, \alpha_z)$ 
7: end function

```

---



**Algorithm 4** verifies a proof

---

```

1: function VERIFYPROOF( $id, u_t, \alpha_z, u$ )
2:    $c' \leftarrow \mathcal{H}_0(g, u_t, u, id)$ .
3:    $u_z \leftarrow g^{\alpha_z}$ .
4:   if  $u_z = u_t u^{c'}$  then
5:     return 1
6:   else
7:     return 0
8:   end if
9: end function

```

---

**3.2.4. Digital Signatures**

A digital signature scheme SS consists of the following algorithms [32].

- $G$  is a probabilistic algorithm that takes a security parameter. It outputs a pair  $(vk, sk)$ , where  $sk$  is a secret signing key, and  $vk$  is a public verification key.
- $sign$  is a probabilistic algorithm that is invoked as  $\sigma \leftarrow sign(sk, m)$ , where  $sk$  is a secret key (as output by the key generation algorithm) and  $m$  is a message. The algorithm outputs a signature  $\sigma$ .
- $verify$  is a deterministic algorithm invoked as  $b \leftarrow verify(vk, m, \sigma)$ , where  $b$  is a bit. If  $b = 1$ , then it means the signature is accepted, or else it is rejected.

**3.3. Threat Model**

We assume a semi-honest adversary, i.e., one who corrupts parties but follows the protocol as specified. Under this threat model, the corrupt parties follow the rules of the protocol honestly but they may attempt to learn as much as possible from the messages they receive from other parties to control the creation of blocks in the chain. Furthermore, there may be several colluding corrupt parties combining their partial views to learn information. Semi-honest adversaries are regarded as passive since they do not take any active actions other than attempting to learn private information by observing a view of the protocol execution. Semi-honest adversaries are also commonly called honest-but-curious.

We regard the view of a party as its private inputs, its memory data, and the list of all messages received during the protocol. In this sense, the view of an adversary is composed of the combined views of all corrupt parties. Therefore, under this threat model, any information the adversary learns from the run of the protocol must be a computable function on the input of its combined view [33].

**3.4. Initial Assumptions**

We assume that each participant has access to a long-term key pair  $(sk, vk)$  generated by a signature scheme SS. Furthermore, each participant has access to a digital certificate that proves the validity of the corresponding public verification key  $vk$ . Additionally, when two participants want to communicate, a secure channel is established between them via a protocol such as transport layer security (TLS) [32].

**3.5. Initial Design**

Here, we present our first attempt to build a proof of accuracy protocol. In particular, this initial design is a proof-of-concept based on the cryptographic components required to compose a proof-of-accuracy (PoAc) protocol described by [18]. Specifically, [18] presents the main cryptographic constituents to build such a protocol, but they do not present a concrete cryptographic construction of the protocol.

Let us assume that there are  $m + 1$  participants. One of them assumes the role of coordinator with identifier 0. Additionally, each  $m$  remaining participant is given a unique identifier  $i \in \mathbb{ID}_m$ . The arithmetic works over a cyclic group  $\mathbb{G} \subseteq \mathbb{Z}_p$  generated by  $g$  and whose order is a prime number  $q$  with  $p = 2q + 1$  ( $p$  a prime number). The initial design runs as follows.

1. Participant  $i$  generates an ephemeral key pair by selecting the private key  $\kappa_i \in \mathbb{Z}_q$  at random and computing the public key  $pk_i \leftarrow g^{\kappa_i} \in \mathbb{G}$
2. Participant  $i$  sends the ephemeral public key  $pk_i$  to the other  $m - 1$  participants.
3. Once participant  $i$  receives other participants' ephemeral public keys, the participant computes  $R_i$  as  $R_i = \sum_{j=1, j \neq i}^m s_i(j) pk_j^{\kappa_j}$ , where  $s$  is the function defined in Equation (1).
4. Each participant  $i$  selects  $c_i \in \mathbb{Z}_q$  at random and computes  $\gamma_i = c_i + R_i$ . The participant then sends its  $\gamma_i$  to the coordinator.
5. Once the coordinator receives  $\gamma_i$ 's from all participants, the coordinator will compute the secret  $\alpha$  as

$$\alpha = \sum_{i=1}^m \gamma_i = \sum_{i=1}^m c_i + \sum_{i=1}^m R_i = \sum_{i=1}^m c_i,$$

since  $\sum_{i=1}^m R_i = 0$ .

6. The coordinator generates  $s$  shares of  $\alpha$ ,  $(\alpha_1, \dots, \alpha_s) \leftarrow G_{sh}(s, t, \alpha)$ , and  $n - s$  random points  $\alpha_{s+1}, \alpha_{s+2}, \dots, \alpha_n$  from  $\mathbb{Z}_q \times \mathbb{Z}_q$ . The coordinator then computes  $u = g^\alpha$  and shuffles the genuine and non-genuine points, forming the list  $A = [\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_n}]$ , with  $1 \leq i_1, i_2, \dots, i_n \leq n$ . The coordinator now computes  $\sigma_0 = \text{sign}(\text{sk}_0, \mathcal{H}_1(A \parallel u \parallel B_l))$  where  $B_l$  is the last block in the blockchain. The coordinator now makes  $A, u, \sigma_0$  publicly accessible to all participants.
7. At this point, the mining process begins. A mining party will attempt to reconstruct the secret  $\alpha$  by finding  $t$  genuine points from  $A$ . In particular, the participant first collects  $A, u, \sigma_0$  from the coordinator, and may check whether  $\sigma_0$  is a valid signature for  $\mathcal{H}_1(A \parallel u \parallel B_l)$ . The participant then selects  $t$  points  $\alpha_1, \dots, \alpha_t$ , computes  $\alpha' \leftarrow C_{sh}(\alpha_1, \dots, \alpha_t)$ , and checks whether  $u$  is equal to  $g^{\alpha'}$ . Once the participant finds  $t$  suitable points, the participant will proceed with step 8. Otherwise, the participant will attempt to find  $t$  genuine points.
8. A mining party with identifier  $id$  proves its knowledge of the correct  $\alpha'$  to other participants by using the Schnorr non-interactive zero-knowledge scheme. Specifically,
  - (a) The participant computes  $\text{proof} = (id, u_t, \alpha_z) \leftarrow \text{genProof}(\alpha', u, id)$ . He then publishes  $(m_{id}, \sigma_{id})$  to the network, where

$$m_{id} \leftarrow (\text{proof}, A, u, \sigma_0)$$

and  $\sigma_{id} \leftarrow \text{sign}(\text{sk}_{id}, \mathcal{H}_1(m_{id}))$ .

- (b) Any verifier can check a solution  $(m_{id}, \sigma_{id})$  by calling the function check shown by Algorithm 5.

---

#### Algorithm 5 checks a solution

---

```

1: function CHECK( $m_{id}, \sigma_{id}, \text{vk}_{id}, \text{vk}_0, B_l$ )
2:    $b_0 \leftarrow \text{verify}(\text{vk}_{id}, \mathcal{H}_1(m_{id}), \sigma_{id})$ 
3:   if  $b_0 = 1$  then
4:      $(\text{proof}, A, u, \sigma_0) \leftarrow m_{id}$ 
5:      $b_1 \leftarrow \text{verify}(\text{vk}_0, \mathcal{H}_1(A \parallel u \parallel B_l), \sigma_0)$ 
6:      $(id, u_t, \alpha_z) \leftarrow \text{proof}$ 
7:      $b_2 \leftarrow \text{verifyProof}(id, u_t, \alpha_z, u)$ 
8:     if  $b_1 = 1$  and  $b_2 = 1$  then
9:       return Accept
10:    else
11:      return Reject
12:    end if
13:  else
14:    return Reject
15:  end if
16: end function

```

---



### 3.5.1. Analysis of the Initial Design

Here, we analyze the initial design.

#### Correctness

We analyze step 7 of the initial design. Note that what the coordinator does is to call  $G_{sh}(s, t, \alpha)$ , creating  $s$  genuine shares for  $\alpha$ , any  $t$  of which serves to reconstruct  $\alpha$ . Hence, any mining party that finds  $t$  genuine shares among the  $n$  entries in  $A$  will successfully reconstruct the secret.

#### Drawbacks

The major drawback of the initial design is that it heavily relies on the coordinator since this coordinator aggregates all  $\gamma_i$  to obtain the secret  $\alpha$  and then computes the  $s$  shares of  $\alpha$  and  $n - s$  random points, which means the coordinator gains too much knowledge of  $\alpha$  and, hence, may take advantage of this knowledge to favor any mining party.

Ideally, this coordinator must not know  $\alpha$ , neither the  $s$  genuine shares of  $\alpha$  nor the  $n - s$  random points, i.e., the coordinator only should serve as an aggregator of data. The following design improves upon the initial one by exploiting further the aggregating protocols [30,31] to compute the genuine and non-genuine shares securely.

### 3.6. An Improved Design

Let us assume that there are  $t$  participants. One of them assumes the role of the coordinator with identifier 0. Each participant other than the coordinator has a unique identifier  $i \in \mathbb{ID}_{t-1}$ . The arithmetic works over a cyclic group  $\mathbb{G} \subseteq \mathbb{Z}_p$  generated by  $g$ , whose order is a prime number  $q$  with  $p = 2q + 1$  ( $p$  a prime number). The improved design runs as follows:

1. Participant  $i$  generates  $n + 1$  ephemeral key pairs

$$(\kappa_{i,0}, g^{\kappa_{i,0}}), (\kappa_{i,1}, g^{\kappa_{i,1}}), \dots, (\kappa_{i,n}, g^{\kappa_{i,n}})$$

by randomly selecting  $\kappa_{i,k} \in \mathbb{Z}_q$  and computing  $g^{\kappa_{i,k}}$  for each  $k \in \{0, 1, \dots, n\}$ .

2. Participant  $i$  sends its ephemeral public keys

$$(g^{\kappa_{i,0}}, g^{\kappa_{i,1}}, g^{\kappa_{i,2}}, \dots, g^{\kappa_{i,n}})$$

to all other participants.

3. After receiving the ephemeral public keys from each other participant, the participant  $i$  computes the following vector

$$R_i = (R_{i,0}, R_{i,1}, \dots, R_{i,n}),$$

where

$$R_{i,0} = \prod_{j=1, j \neq i}^{t-1} (g^{\kappa_{j,0}})^{s_i(j)\kappa_{i,0}}$$

and

$$R_{i,k} = \sum_{j=1, j \neq i}^{t-1} s_i(j)(g^{\kappa_{j,k}})^{\kappa_{i,k}}$$

for each  $k \in \{1, 2, \dots, n\}$ .

4. Participant  $i$  selects  $\gamma_i, c_i \xleftarrow{R} \mathbb{Z}_q$  and a probability  $p_i \xleftarrow{R} [p_{min}, 1]$ . This participant computes the vector  $C_i = (g^{\gamma_i} R_{i,0}, e_{i,1} + R_{i,1}, \dots, e_{i,n} + R_{i,n})$ , where

$$e_{i,k} = \begin{cases} c_i k^i + \gamma_i \bmod q & \text{with probability } p_i \\ z \xleftarrow{R} \mathbb{Z}_q & \text{with probability } 1 - p_i \end{cases}$$

and sends  $C_i$  to the coordinator

5. Once the coordinator receives each  $C_i$  for  $i = 1, \dots, t-1$ , the coordinator will compute

$$A = \left( \prod_{i=1}^{t-1} C_{i,0}, \sum_{i=1}^{t-1} C_{i,1}, \dots, \sum_{i=1}^{t-1} C_{i,n} \right) \quad (2)$$

The coordinator now computes  $\sigma_0 = \text{sign}(\text{sk}_0, \mathcal{H}_1(A \parallel B_l))$ , where  $B_l$  is the last block in the blockchain. The coordinator now makes  $A, \sigma_0$  publicly accessible to all participants.

6. At this point, the mining process begins. A mining party first collects  $A$  and  $\sigma_0$  from the coordinator, and may check whether  $\sigma_0$  is a valid signature for  $\mathcal{H}_1(A \parallel B_l)$ . It then will attempt to find  $t$  unique indices  $1 \leq k_1, k_2, \dots, k_t \leq n$ , such that  $A_0 = g^{\alpha'}$ , where

$$\alpha' \leftarrow \mathcal{C}_{sh}((k_1, A_{k_1}), (k_2, A_{k_2}), (k_3, A_{k_3}), \dots, (k_t, A_{k_t})).$$

Once the participants find  $t$  suitable points, they will proceed with step 7. Otherwise, they will attempt to find  $t$  genuine points.

7. A mining party with identifier  $id$  proves its knowledge of the correct  $\alpha'$  to other participants by using the Schnorr non-interactive zero-knowledge scheme. Specifically,
- The participant computes  $\text{proof} = (id, u_t, \alpha_z) \leftarrow \text{genProof}(\alpha', A_0, id)$ . He then publishes  $(m_{id}, \sigma_{id})$  to the network, where  $m_{id} \leftarrow (\text{proof}, A, \sigma_0)$  and  $\sigma_{id} \leftarrow \text{sign}(\text{sk}_{id}, \mathcal{H}_1(m_{id}))$ .
  - Any verifier can check a solution  $(m_{id}, \sigma_{id})$  by calling the function `check` shown by Algorithm 6.

---

**Algorithm 6** checks a solution

---

```

1: function CHECK( $m_{id}, \sigma_{id}, \text{vk}_{id}, \text{vk}_0, B_l$ )
2:    $b_0 \leftarrow \text{verify}(\text{vk}_{id}, \mathcal{H}_1(m_{id}), \sigma_{id})$ 
3:   if  $b_0 = 1$  then
4:      $(\text{proof}, A, \sigma_0) \leftarrow m_{id}$ 
5:      $b_1 \leftarrow \text{verify}(\text{vk}_0, \mathcal{H}_1(A \parallel B_l), \sigma_0)$ 
6:      $(id, u_t, \alpha_z) \leftarrow \text{proof}$ 
7:      $b_2 \leftarrow \text{verifyProof}(id, u_t, \alpha_z, A_0)$ 
8:     if  $b_1 = 1$  and  $b_2 = 1$  then
9:       return Accept
10:    else
11:      return Reject
12:    end if
13:  else
14:    return Reject
15:  end if
16: end function
```

---

### 3.6.1. Analysis of the Improved Design

Here, we analyze the improved design.

#### Correctness

We analyze steps 5 and 6 of the improved design. Let us assume that each  $C_i$  obtained from participant  $i$  was created with a fixed probability  $p_i$ . Note that since  $\prod_{i=1}^{t-1} R_{i,0} = 1$ ,

$$A_0 = \prod_{i=1}^{t-1} C_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} R_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} \prod_{i=1}^{t-1} R_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} = g^{\sum_{i=1}^{t-1} \gamma_i} = g^{\alpha} = u$$

where  $\alpha = \sum_{i=1}^{t-1} \gamma_i \bmod q$ .

Let us fix a  $k$  with  $1 \leq k \leq n$ . Let us assume that  $C_{i,k} = c_i k^i + \gamma_i + R_{i,k} \bmod q$  for all  $1 \leq i \leq t-1$ . By construction, this occurs with probability  $\rho = p_1 p_2 \dots p_{t-1}$ . Since  $\sum_{i=1}^{t-1} R_{i,k} = 0$ , then

$$A_k = \sum_{i=1}^{t-1} C_{i,k} = \sum_{i=1}^{t-1} c_i k^i + \sum_{i=1}^{t-1} \gamma_i + \sum_{i=1}^{t-1} R_{i,k} = P(k)$$

where  $P(x) = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \dots + c_1x + \alpha$ .

If the mining party finds  $t$  suitable  $k_1, k_2, \dots, k_t$ , then

$$A_{k_1} = P(k_1), A_{k_2} = P(k_2) \dots, A_{k_t} = P(k_t).$$

Therefore,  $A_0 = g^{\alpha'}$ , where

$$\alpha' = \mathcal{C}_{sh}((k_1, A_{k_1}), (k_2, A_{k_2}), \dots, (k_t, A_{k_t})).$$

### Drawbacks

The improved design still relies on a coordinator, but now this coordinator does not know  $\alpha$ , neither the  $s$  genuine shares of  $\alpha$  nor the  $n - s$  random points, i.e., the coordinator only serves as an aggregator of data; if the coordinator wants to know the secret  $\alpha$ , it will have to perform step 6 of the improved design as any other mining party would. However, having a coordinator still presents a unique point of failure for this approach; Also, it solely relies on the proof of work (which may not have a solution) performed by a mining party at step 6. To reduce power concentration (hence, the 51% attack [5]) and increase the fairness of the mining process, a new version should complement the proof of work with other approaches, for example, access to random locations (similar to PoC) [17].

Another drawback is that any mining party will attempt to recover  $\alpha$  from  $A$  at step 6. Ideally, any mining party should only know how to reconstruct  $g^\alpha$  rather than  $\alpha$ . Another issue may arise if the cyclic group  $\mathbb{G}$  is set to another one (e.g., a subgroup of the points of an elliptic curve). If that is the case, a mapping to associate each element of  $\mathbb{G}$  with an element of  $\mathbb{Z}_q$  will be required. In this version, this is not a problem since we are assuming the arithmetic works over a cyclic group  $\mathbb{G} \subseteq \mathbb{Z}_p$  generated by  $g$  with order  $q$ , where  $p = 2q + 1$  and  $p, q$  are prime numbers. Hence, the computation of

$$R_{i,k} = \sum_{j=1, j \neq i}^{t-1} s_i(j) (g^{k_{j,k}})^{k_{i,k}}$$

for each  $k \in \{1, 2, \dots, n\}$  does not present any inconvenient.

Our final version deals with the drawbacks of the improved design by introducing the following features:

- Permit a mining party to access  $C_i$  at different locations and compute a new vector  $A$  independently with the collected  $C_i$ 's.
- Use only the multiplicative version of the aggregating protocol [31] to compute the ciphertexts of the shares.
- Exploit the homomorphic properties of ElGamal-based cryptographic schemes to allow any mining party to reconstruct  $g^\alpha$  from the ciphertexts of the shares.

### 3.7. Our Proposed Protocol

Let us assume that there are  $t - 1$  participants. Each participant has a unique identifier  $i \in \mathbb{D}_{t-1}$ . The arithmetic works over a cyclic group  $\mathbb{G}$  generated by  $g$  and whose order is a prime number  $q$ . Our proposed protocol runs as follows.

1. Participant  $i$  generates  $n + 1$  ephemeral key pairs

$$(\kappa_{i,0}, g^{\kappa_{i,0}}), (\kappa_{i,1}, g^{\kappa_{i,1}}), \dots, (\kappa_{i,n}, g^{\kappa_{i,n}})$$

- by selecting  $\kappa_{i,k} \in \mathbb{Z}_q$  at random and computing  $g^{\kappa_{i,k}}$  for each  $k \in \{0, 1, \dots, n\}$ .
2. Participant  $i$  sends its ephemeral public keys

$$(g^{\kappa_{i,0}}, g^{\kappa_{i,1}}, g^{\kappa_{i,2}}, \dots, g^{\kappa_{i,n}})$$

to all other participants.

3. After receiving the ephemeral public keys from the other  $t - 2$  participants, participant  $i$  computes the following vector

$$R_i = (R_{i,0}, R_{i,1}, \dots, R_{i,n}),$$

where

$$R_{i,k} = \prod_{j=1, j \neq i}^{t-1} (g^{\kappa_{j,k}})^{s_i(j)\kappa_{i,k}}$$

for each  $k \in \{0, 1, \dots, n\}$

4. At this point, the mining process begins. A mining party will have to contact each participant  $i$  and request a  $C_i$  from it. Specifically, upon request, the participant  $i$  executes  $(C_i, \sigma_i) \leftarrow \text{generateC}(\text{sk}_i, R_i, B_l)$ , shown by Algorithm 7, where  $B_l$  is the last block in the blockchain.

---

**Algorithm 7** generates the pair  $(C_i, \sigma_i)$

---

```

1: function GENERATEC( $\text{sk}_i, R_i, B_l$ )
2:    $p_i \xleftarrow{R} [p_{\min}, 1]$ .
3:    $c_i \xleftarrow{R} \mathbb{Z}_q$ 
4:    $\gamma_i \xleftarrow{R} \mathbb{Z}_q$ 
5:    $C_{i,0} \leftarrow g^{\gamma_i} R_{i,0}$ 
6:   for  $k \leftarrow 1$  to  $n$  do
7:      $p \xleftarrow{R} [0, 1]$ 
8:     if  $p \leq p_i$  then
9:        $e_{i,k} \leftarrow c_i k^i + \gamma_i \bmod q$ 
10:    else
11:       $e_{i,k} \xleftarrow{R} \mathbb{Z}_q$ 
12:    end if
13:     $C_{i,k} \leftarrow g^{e_{i,k}} R_{i,k}$ 
14:  end for
15:   $\sigma_i \leftarrow \text{sign}(\text{sk}_i, \mathcal{H}_1(C_i \parallel B_l))$ .
16:  return  $(C_i, \sigma_i)$ 
17: end function

```

---

The participant  $i$  then sends  $(C_i, \sigma_i)$  to the requesting mining party. Note that the mining party may check whether  $\sigma_i$  is a valid signature for  $\mathcal{H}_1(C_i \parallel B_l)$ . Once the mining party contacts each participant  $i$  and collects the corresponding  $C_i$ , the party then computes  $A$  as

$$A = \left( \prod_{i=1}^{t-1} C_{i,0}, \prod_{i=1}^{t-1} C_{i,1}, \dots, \prod_{i=1}^{t-1} C_{i,n} \right).$$

The mining party's goal is to find unique indices  $1 \leq k_1, k_2, \dots, k_t \leq n$ , such that

$$A_0 = w$$

with

$$w = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots (A_{k_t})^{\lambda_{k_t}}$$

and

$$\lambda_{k_j} = \prod_{\substack{r=1 \\ r \neq j}}^t \frac{-k_r}{k_j - k_r} \text{ for } 1 \leq j \leq t.$$

Once the participant finds  $t$  unique indices, the participant will proceed with step 5. Otherwise, this mining party may attempt step 4 again.

5. When a mining party with identifier  $id$  reconstructs  $w$ , i.e., finds suitable unique indices  $1 \leq k_1, k_2, \dots, k_t \leq n$ , it will publish  $(m_{id}, \sigma_{id})$  to the network, where

$$m_{id} = (id, k_1, k_2, \dots, k_t, (C_1, \sigma_1), (C_2, \sigma_2), \dots, (C_{t-1}, \sigma_{t-1})),$$

and

$$\sigma_{id} = \text{sign}(\text{sk}_{id}, \mathcal{H}_1(m_{id})).$$

6. Any verifier can check a solution  $(m_{id}, \sigma_{id})$  by calling the function check shown by Algorithm 8.

---

**Algorithm 8** checks a solution

---

```

1: function CHECK( $m_i, \sigma_{id}, \text{vk}_{id}, \text{vk}_1, \text{vk}_2, \dots, \text{vk}_{t-1}, B_l$ )
2:    $b_0 \leftarrow \text{verify}(\text{vk}_{id}, \mathcal{H}_1(m_{id}), \sigma_{id})$ 
3:   if  $b_0 = 1$  then
4:      $(id, k_1, k_2, \dots, k_t, (C_1, \sigma_1), (C_2, \sigma_2), \dots, (C_{t-1}, \sigma_{t-1})) \leftarrow m_{id}$ 
5:     for  $i \leftarrow 1$  to  $t - 1$  do
6:       if  $\text{verify}(\text{vk}_i, \mathcal{H}_1(C_i \parallel B_l), \sigma_i) = 0$  then
7:         return Reject
8:       end if
9:     end for
10:    Compute

$$A = \left( \prod_{i=1}^{t-1} C_{i,0}, \prod_{i=1}^{t-1} C_{i,1}, \dots, \prod_{i=1}^{t-1} C_{i,n} \right).$$

11:    With the given indices  $1 \leq k_1, k_2, \dots, k_t \leq n$ , compute  $\lambda_{k_j} = \prod_{\substack{r=1 \\ r \neq j}}^t \frac{-k_r}{k_j - k_r}$  for  $1 \leq j \leq t$ .
12:    Compute  $w = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots (A_{k_t})^{\lambda_{k_t}}$ 
13:    if  $w = A_0$  then
14:      return Accept
15:    else
16:      return Reject
17:    end if
18:  else
19:    return Reject
20:  end if
21: end function

```

---

#### 4. Protocol Analysis

In this section, we will make a deeper analysis of our proposed protocol.

##### 4.1. Correctness of Our Proposed Protocol

We now analyze step 4 of our proposed protocol. Let us assume that each  $C_i$  obtained from participant  $i$  was created with a fixed probability  $p_i$ . Note that  $\prod_{i=1}^{t-1} R_{i,k} = 1$  for any  $0 \leq k < n$ , then

$$A_0 = \prod_{i=1}^{t-1} C_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} R_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} \prod_{i=1}^{t-1} R_{i,0} = \prod_{i=1}^{t-1} g^{\gamma_i} = g^{\sum_{i=1}^{t-1} \gamma_i} = g^\alpha = u$$

where  $\alpha = \sum_{i=1}^{t-1} \gamma_i \bmod q$ .

Let us fix a  $k$  with  $1 \leq k \leq n$ . Let us assume that  $C_{i,k} = g^{e_{i,k}} R_{i,k}$  with  $e_{i,k} = c_i k^i + \gamma_i \bmod q$  for all  $1 \leq i \leq t-1$ . By construction, this occurs with probability  $\rho = p_1 p_2 \dots p_{t-1}$ . Therefore,

$$A_k = \prod_{i=1}^{t-1} C_{i,k} = \prod_{i=1}^{t-1} g^{e_{i,k}} R_{i,k} = \prod_{i=1}^{t-1} g^{e_{i,k}} \prod_{i=1}^{t-1} R_{i,k} = g^{\sum_{i=1}^{t-1} c_i k^i + \sum_{i=1}^{t-1} \gamma_i} = g^{P(k)}$$

where  $P(X) = c_{t-1}X^{t-1} + c_{t-2}X^{t-2} + \dots + c_1X + \alpha$  with  $\alpha = \sum_{i=1}^{t-1} \gamma_i$ .  
If the mining party finds  $t$  suitable  $k_1, k_2, \dots, k_t$ , then

$$A_{k_1} = g^{P(k_1)}, A_{k_2} = g^{P(k_2)} \dots, A_{k_t} = g^{P(k_t)}.$$

Hence,

$$A_0 = g^\alpha = g^{P(0)} = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots \cdot (A_{k_t})^{\lambda_{k_t}},$$

where

$$\lambda_{k_j} = \prod_{\substack{r=1 \\ r \neq j}}^t \frac{-k_r}{k_j - k_r} \text{ with } 1 \leq j \leq t.$$

In Section 4.2, we extend our analysis of this mining process.

#### 4.2. Mining Process

We here further analyze step 4 of our proposed protocol.

##### 4.2.1. Estimating $n$ and $s$

Let us suppose  $t$  is fixed. Let  $C_i$  be the array obtained from participant  $i$  at step 4 of our proposed protocol for  $1 \leq i \leq t-1$ . Assume that  $C_{i,k}$  for  $1 \leq k \leq n$  has been created with probability  $p_i$  by the participant  $i$ . Note that the value  $A_k$  depends on  $C_{i,k}$  for  $1 \leq i \leq t-1$ . Therefore,  $\rho = \prod_{i=1}^{t-1} p_i$  is the probability of obtaining a genuine entry  $A_k$  in  $A$ . If we define the random variable  $X$  as the number of genuine  $A_k$ 's in the sequence of values  $A_1, A_2, A_3, \dots, A_n$ , then  $X$  follows a binomial distribution with success probability  $\rho$  in the sequence of  $n$  independent trials. Therefore, the expected number of genuine  $A_k$ 's is given by  $E[X] = n \cdot \rho$ . By choosing  $n$ , such that  $E[X] \geq t$ ,  $s$  is expected to be greater than  $t$ . Since  $p_{\min}$  is known and  $0 \leq p_{\min} \leq p_1, p_2, \dots, p_{t-1} \leq 1$ , then  $n \cdot p_{\min}^{t-1} \leq n \cdot \rho = E[X]$ , and so

$$E[X] \geq s_{\min} = n \cdot p_{\min}^{t-1} \geq t$$

when  $n \geq \frac{t}{p_{\min}^{t-1}}$ .

##### 4.2.2. Expected Number of Attempts for Recovering $A_0$

We want to estimate the expected number of attempts to solve the proof of work assuming that the number of genuine  $A_i$ 's is at least  $t$ , i.e.,  $s \geq t$  (otherwise, the PoW cannot be solved). Let  $\mathcal{Z}$  be the set of all combinations  $[k_1, k_2, \dots, k_t]$  out of  $[1, 2, \dots, n]$ .

A first strategy by the miner is to select  $\Omega \subseteq \mathcal{Z}$  and call  $\text{mining}_0(\Omega, A)$  as shown by Algorithm 9. We call  $\omega$  a genuine combination if the reconstructed  $w$  is equal to  $A_0$  (i.e., line 7 of the function  $\text{mining}_0$  satisfies). Otherwise,  $\omega$  is a non-genuine combination.

---

**Algorithm 9** describes the first mining strategy

---

```

1: function MINING0(Ω, A)
2:   while True do
3:     ω ←R Ω
4:     [k1, k2, ..., kt] ← ω
5:     Compute λkj = ∏r=1, r≠jt  $\frac{-k_r}{k_j - k_r}$  for 1 ≤ j ≤ t.
6:     Compute w = (Ak1)λk1 · (Ak2)λk2 · ... · (Akt)λkt
7:     if w = A0 then
8:       return ω
9:     end if
10:   end while
11: end function

```

---



Let  $N = |\Omega|$  be the total number of combinations in  $\Omega$ . Let  $K_1$  be the number of genuine combinations in  $\Omega$ . Therefore,  $K_2 = N - K_1$  is the number of non-genuine combinations in  $\Omega$ . When  $\Omega = \mathcal{Z}$ ,  $N = \binom{n}{t}$  and  $K_1 = \binom{s}{t}$ .

Let  $Y_0$  be the random variable that counts the number of iterations to find a genuine combination in  $\Omega$  using the strategy  $\text{mining}_0$ . Therefore,  $Y_0$  follows a geometric distribution with success probability  $\frac{K_1}{N}$ . Hence,

$$\Pr[Y_0 = y_0] = (1 - \frac{K_1}{N})^{y_0-1} \frac{K_1}{N}$$

for  $y_0 = 1, 2, 3, \dots$  and  $E[Y_0] = N/K_1$ .

Note that at any iteration of  $\text{mining}_0$ , nothing is learned about a particular  $A_i$ , i.e., it does not learn whether  $A_i$  is genuine or non-genuine, but instead, it does learn whether a particular combination  $\omega$  is genuine or not.

A seemingly better strategy is to pick  $\Omega \subseteq \mathcal{Z}$  and call  $\text{mining}_1(\Omega, A)$  as shown by Algorithm 10. Furthermore, if the mining party has access to computational resources, the party then may make a partition of  $\mathcal{Z}$ , i.e., pick  $\Omega_1, \Omega_2, \dots, \Omega_{n_p}$  with  $\Omega_i \cap \Omega_j = \emptyset$  for  $1 \leq i, j \leq n_p$  with  $i \neq j$ , and  $\mathcal{Z} = \bigcup_{i=1}^{n_p} \Omega_i$ ; and sets and runs  $n_p$  parallel processing tasks, where the processing task  $i$  searches over  $\Omega_i$ , viz. executes  $\text{mining}_1(\Omega_i, A)$ .

---

**Algorithm 10** describes the second mining strategy strategy

---

```

1: function MINING1( $\Omega, A$ )
2:   for each  $\omega \in \Omega$  do
3:      $[k_1, k_2, \dots, k_t] \leftarrow \omega$ 
4:     Compute  $\lambda_{k_j} = \prod_{\substack{r=1 \\ r \neq j}}^t \frac{-k_r}{k_j - k_r}$  for  $1 \leq j \leq t$ .
5:     Compute  $w = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots (A_{k_t})^{\lambda_{k_t}}$ 
6:     if  $w = A_0$  then
7:       return  $\omega$ 
8:     end if
9:   end for
10:  return  $\perp$ 
11: end function

```

---

Let  $Y_1$  be the random variable that counts the number of iterations to find a genuine combination in  $\Omega$  using the strategy  $\text{mining}_1$ . According to [34],

$$\Pr[Y_1 = y_1] = \frac{\binom{N-y_1+1}{K_1}}{\binom{N}{K_1}} \frac{K_1}{N - y_1 + 1}$$

for any  $y_1 \in \{1, \dots, K_2 + 1\}$ . Hence,  $E_{Y_1} = E[Y_1] = \sum_{y_1=1}^{K_2+1} y_1 \Pr[Y = y_1] = \frac{N+1}{K_1+1}$ . Since  $K_1 \leq N$ , then  $K_1 N + K_1 \leq K_1 N + N$  and, therefore,  $E_{Y_1} = \frac{N+1}{K_1+1} \leq \frac{N}{K_1} = E_{Y_0}$ .

#### 4.2.3. Experimental Results

Let  $\mathcal{P}$  be the set  $\{0.9, 0.93, 0.95, 0.98, 1\}$  and  $\mathcal{T}$  be the set  $\{20, 30, 40, 50, 60\}$ . We carry out the following experiment, in which a trial consists of the following steps.

1. Choose  $p_{\min} \in \mathcal{P}$ ,  $t \in \mathcal{T}$ . For the selected pair  $(p_{\min}, t)$ , choose a value for  $n$ , such that  $n \geq \frac{t}{p_{\min}^{t-1}}$ .
2. For the selected three-tuple  $(p_{\min}, t, n)$ , perform 100 runs of a modified mining phase (step 5 of our proposed protocol). At each run, the corresponding  $A$  is constructed; the numbers of genuine  $A_i$  (i.e., the value of  $s$ ),  $E[X]$ , and  $s_{\min}$  are computed. Moreover,  $E[Y_0]$  and  $E[Y_1]$  are computed, assuming  $\Omega = \mathcal{Z}$ .
3. At the end of the trial, the means of all values  $s$ ,  $E[X]$ ,  $s_{\min}$ ,  $E[Y_0]$ , and  $E[Y_1]$  are respectively computed.

Figures 2–6 show the results obtained for  $p_{min} \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .

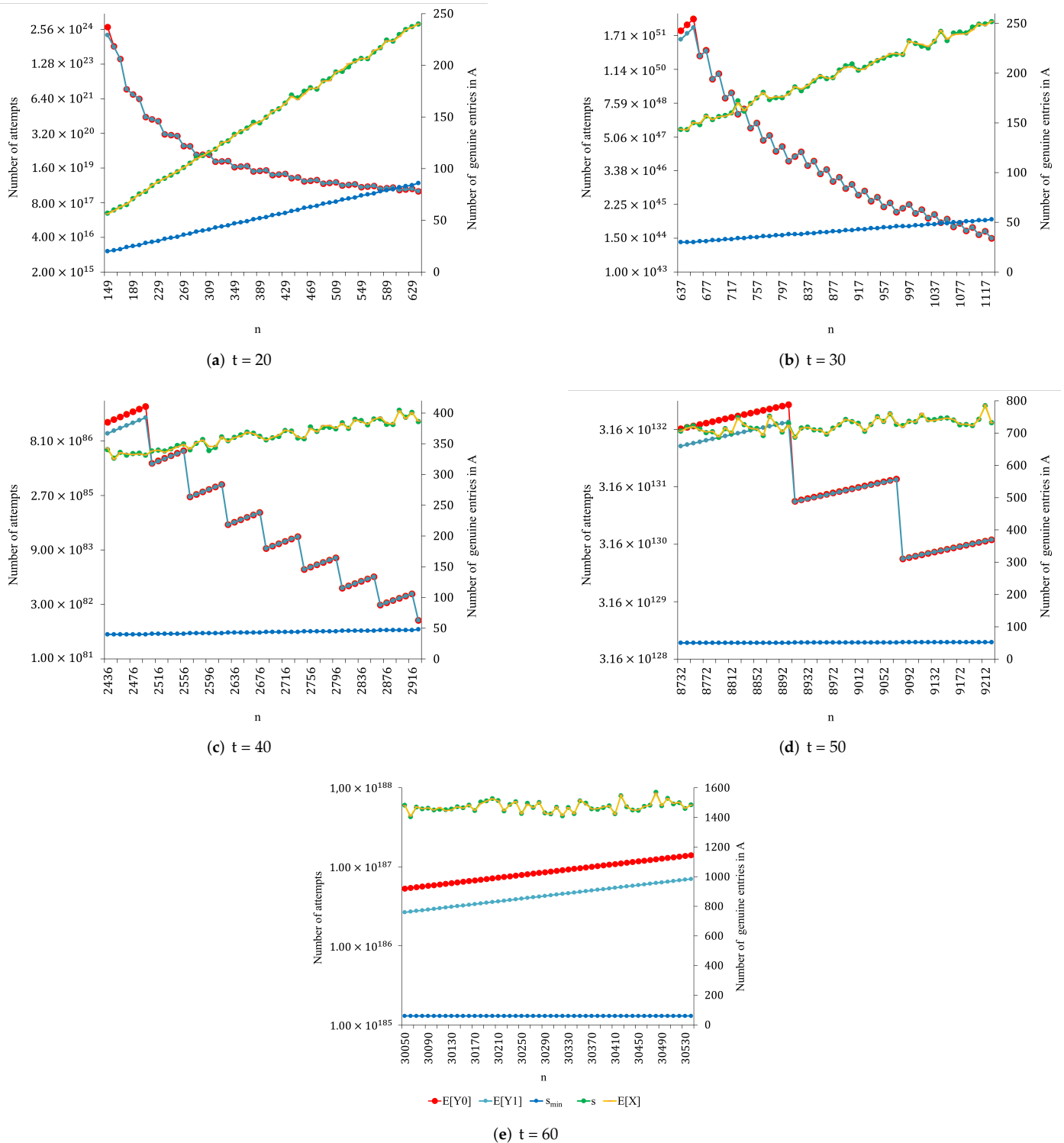


Figure 2. Results obtained for  $p_{min} = 0.90 \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .

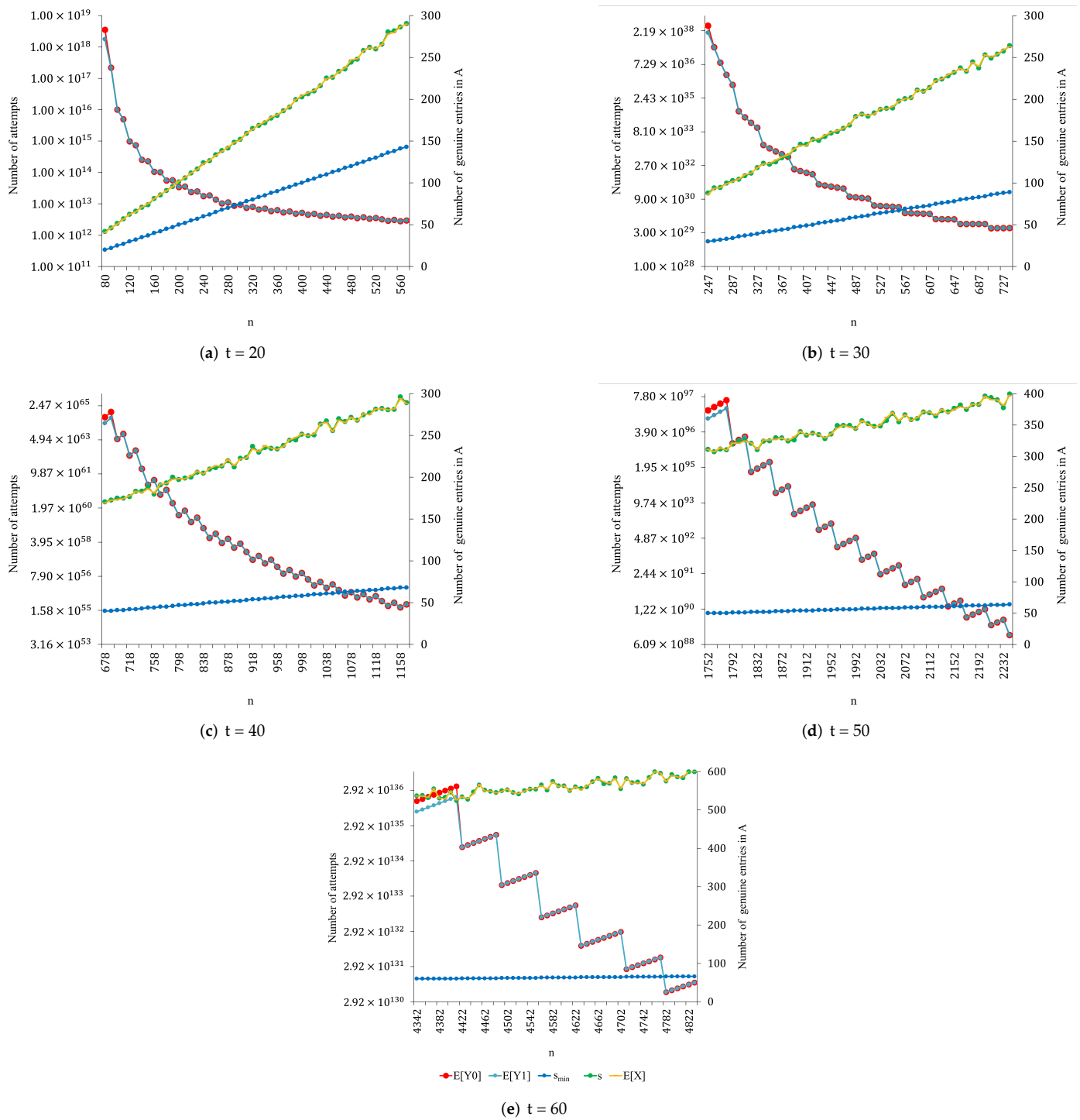


Figure 3. Results obtained for  $p_{\min} = 0.93 \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .

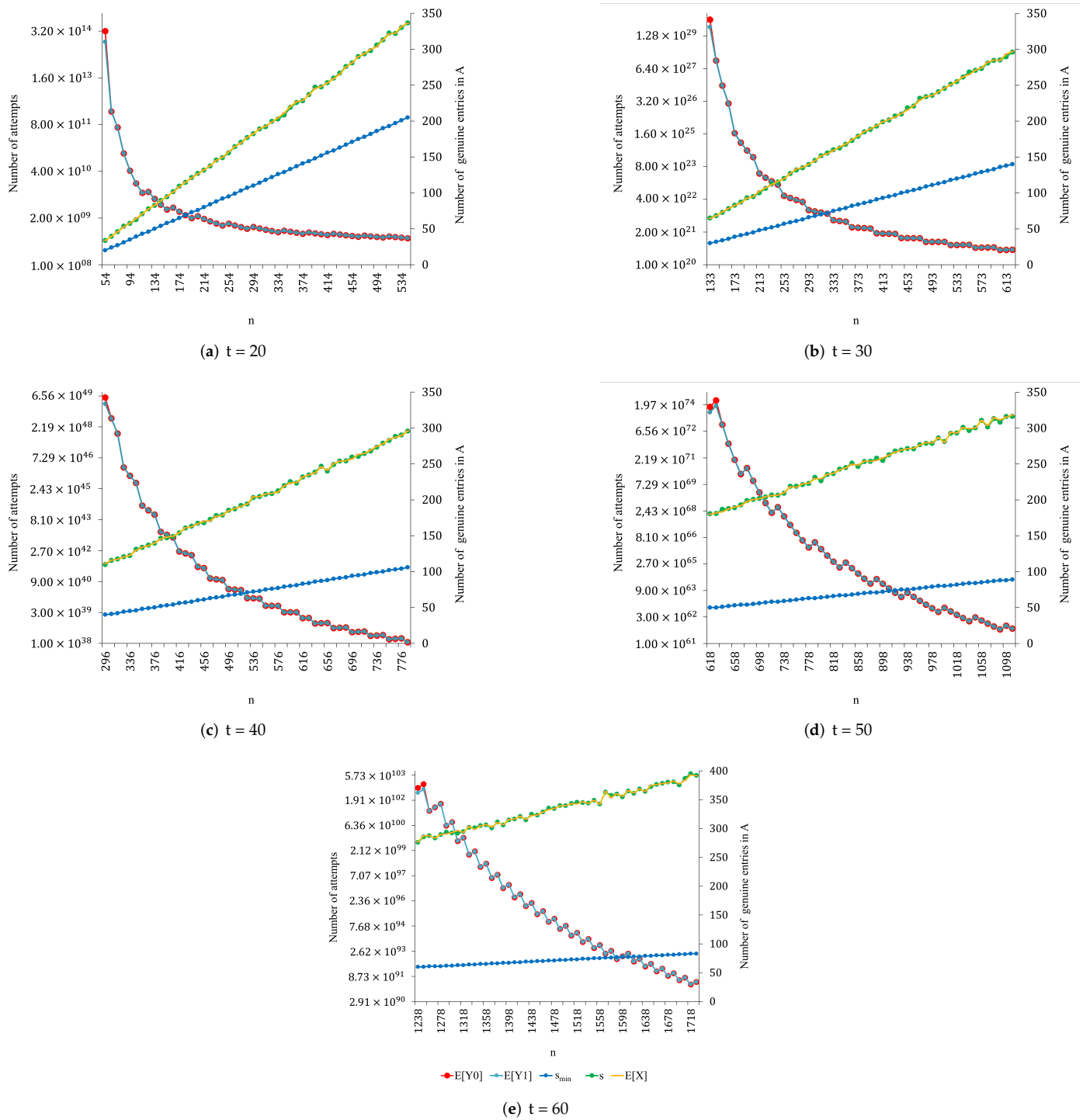


Figure 4. Results obtained for  $p_{\min} = 0.95 \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .

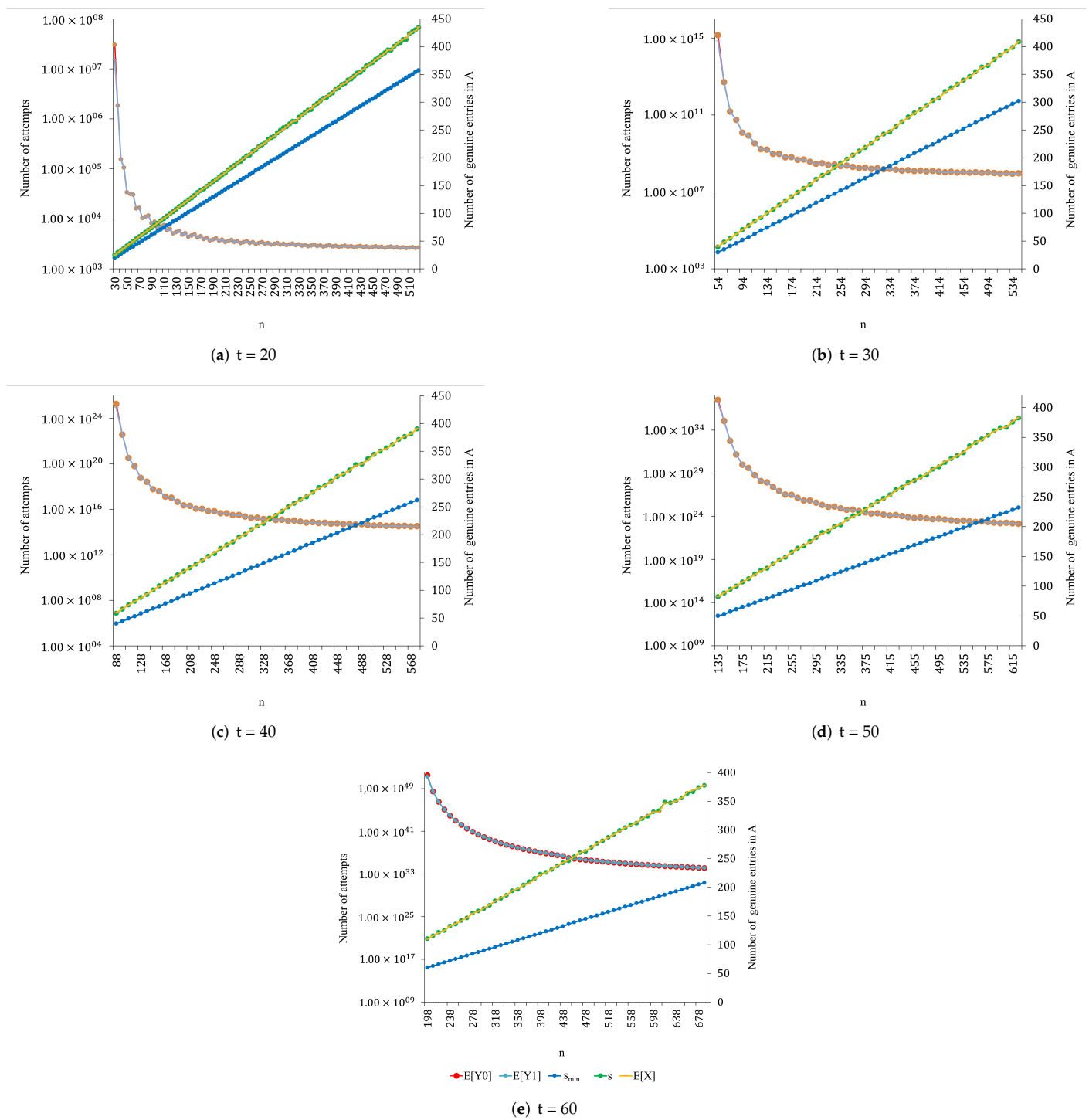
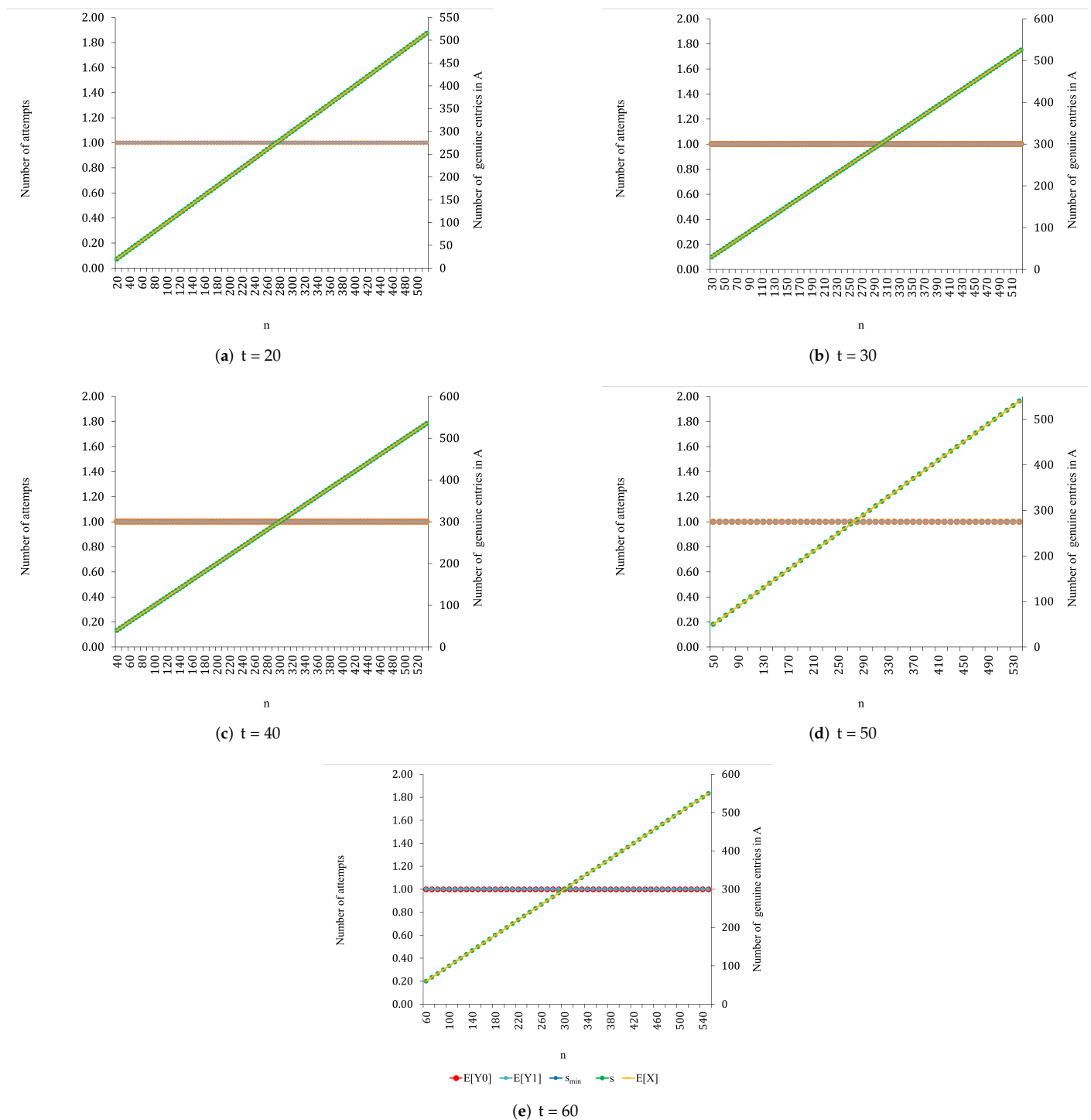


Figure 5. Results obtained for  $p_{\min} = 0.98 \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .



**Figure 6.** Results obtained for  $p_{\min} = 1 \in \mathcal{P}$ ,  $t \in \mathcal{T}$ , and proper values of  $n$ .

### 4.3. Computational Cost Analysis

In this section, we analyze the computation costs related to some critical steps of our protocol. Let  $C_{GO}$  denote the cost of a group operation, and let  $C_{FA}$ ,  $C_{FM}$ , and  $C_{FI}$  denote the costs of a field addition, field multiplication, and field inversion, respectively. We denote by  $C_{GE}$  the cost of an exponentiation in the group, i.e.,  $a^m$  with  $a \in \mathbb{G}$ . Using a generic fast exponentiation algorithm, then  $C_{GE}$  will have a cost of, at most,  $r \cdot C_{GO}$ , where  $r = \lceil \log_2(m) \rceil$ .



#### Computational Costs of Step 4

Once a participant collects all  $C_i$ , the participant needs to compute  $A_k = \prod_{i=1}^{t-1} C_{i,k}$  for  $k = 0, \dots, 1, \dots, n$ . Note that computing  $A_j$  has a cost of  $(t-2)C_{GO}$ ; hence, computing  $A$  has a cost of  $(n+1)(t-2)C_{GO}$ .

Once the participant computes  $A$ , the mining process begins. Until completing the challenge, the participant will keep on selecting  $t$  distinct indices  $1 \leq k_1, k_2, \dots, k_t \leq n$  out of  $\{1, 2, \dots, n\}$ , computing  $w = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots (A_{k_t})^{\lambda_{k_t}}$  with  $\lambda_{k_j} = \prod_{r=1, r \neq j}^t \frac{-k_r}{k_j - k_r}$  for  $1 \leq j \leq t$ , and checking whether  $A_0 = w$ .

So the cost of computing  $w$ , denoted as  $C_w$ , is  $C_w = t \cdot C_{GE} + (t-1) \cdot C_{GO} + t(t-1)(2 \cdot C_{FM} + C_{FI})$ . Therefore, the whole mining process cost is roughly  $E_{Y_1}(C_w + \delta)$  with  $\delta$  being a constant.

#### 4.4. Security Analysis

The goal of the adversary is to gain control over the creation of new blocks in the chain. To that end, he must generate solutions to the proof of works of the protocol and prove their validity to the other participants of the network. Since we assume the adversary is semi-honest, he follows the rules of the protocol but may want to learn as much as possible from the messages he receives from other parties to gain control over the creation of blocks in the chain.

First, note that our proposed protocol in steps 1–3 uses the joint random number generation protocol in parallel. At step 3 of our proposed protocol, what participant  $1 \leq i \leq t-1$  does is to create  $n+1$  ElGamal public keys  $R_{i,k}, 0 \leq k \leq n$  from the other participant's public  $g^{K_{j,k}}$  for  $1 \leq j \leq t-1$ . Since the  $R_{i,k}$  values are constructed from random Diffie–Hellman public-key and cannot be distinguished from random  $R$ 's, the security of this protocol can be reduced to the decisional Diffie–Hellman problem on the group  $\mathbb{G}$  [30]. Furthermore, note that the private key associated with the public key  $R_{i,k}$  is given by  $\sum_{j=1, j \neq i}^{t-1} K_{j,k} \cdot s_i(j) \cdot \kappa_{i,k} \bmod q$  for each  $k \in \{1, 2, \dots, n\}$  and is unknown to any participant, even colluding corrupt parties.

At step 4 of the protocol, a participant, say  $i$ , will send back  $C_i$  with the corresponding signature  $\sigma_i$  to any requesting mining party. Note that each  $C_{i,k}$  is of the form  $g^\epsilon R_{i,k}$  for some  $\epsilon \in \mathbb{Z}_q$ , i.e.,  $C_{i,k}$  represents the ElGamal ciphertext of the message  $g^\epsilon$  under the public key  $R_{i,k}$ .

**Claim 1.** A mining party can obtain a proper and fresh  $A = [A_0, A_1, A_2, \dots, A_n]$  if and only if the mining party has access to the corresponding  $t-1$   $C_i$  and computes

$$A = \left( \prod_{i=1}^{t-1} C_{i,0}, \prod_{i=1}^{t-1} C_{i,1}, \dots, \prod_{i=1}^{t-1} C_{i,n} \right).$$

**Proof.** In a direction (only if), we proved it in Section 4.1. In the other direction, if the requesting mining party has access to  $l_0$  different  $C_{i,k}$  (possibly previous ones) with  $l_0 \neq t-1$ , then the computed  $A_k = \prod_{i=1}^{l_0} C_{i,k}$  will be of the form  $g^\beta$  for some random  $\beta \in \mathbb{Z}_q$ . Hence, the adversary only learns  $g^\beta$  and nothing else.  $\square$

**Claim 2.** A mining party cannot reuse a  $C_i$  for future challenges.

**Proof.** Upon request to participant  $i$  at step 4, a mining party obtains  $(C_i, \sigma_i)$ , where  $\sigma_i \leftarrow \text{sign}(\text{sk}_i, \mathcal{H}_1(C_i \parallel B_l))$ , with  $B_l$  being the last block. Hence, the mining party cannot reuse any  $(C_i, \sigma_i)$  as part of a solution to a future challenge. In particular, by calling the function  $\text{check}()$ , as shown by Algorithm 8, any verifier can discover any cheater.  $\square$

Moreover, the mining process is fair. Let us now assume a mining party (possibly an attacker) constructs a proper and fresh  $A$ , then such a party may start the mining process.

By construction, such a party does not know whether he will find  $t$  suitable  $k_1, k_2, \dots, k_t$  in such  $A$ , with

$$A_{k_1} = g^{P(k_1)}, A_{k_2} = g^{P(k_2)} \dots, A_{k_t} = g^{P(k_t)}.$$

such that

$$A_0 = g^a = g^{P(0)} = (A_{k_1})^{\lambda_{k_1}} \cdot (A_{k_2})^{\lambda_{k_2}} \dots \cdot (A_{k_t})^{\lambda_{k_t}},$$

where

$$\lambda_{k_j} = \prod_{\substack{r=1 \\ r \neq j}}^t \frac{-k_r}{k_j - k_r} \text{ for } 1 \leq j \leq t.$$

If the mining party does not find suitable indices in  $A$ , the mining party may contact any of the  $t - 1$  participants to construct a proper and fresh  $A$  and start the mining process again. Furthermore, the party may request multiple  $C_i$ 's from a participant  $i$ , and then pick only a  $C_i$  from the available ones for each  $i$  to obtain proper and fresh  $A$ 's, and finally search suitable indices in each created  $A$ .

From the previous analysis, we conclude that our proposed protocol can be regarded as secure and fair, assuming a semi-honest adversary.

## 5. Implementation

A proof-of-concept implementation of our proposed protocol was coded in the Python programming language. To simulate the protocol's behavior on a peer-to-peer network, we used the implementation of a decentralized peer-to-peer network [35]. The protocol code was adjusted and some framework classes were modified for the correct implementation of the protocol and its peer-to-peer simulation. After making adjustments to the decentralized peer-to-peer network implementation, we wrote the logic of the proposed protocol in the following python classes:

- Generator class contains methods to create generators for  $\mathbb{G}$ .
- $\mathbb{Z}_q$  class contains methods to carry out operations in  $\mathbb{Z}_q$ .
- MyRsa class contains methods for generating and validating RSA digital signatures.
- Participant class encloses the logic of the proposed protocol. It internally makes calls to P2P methods to send and receive messages over a P2P network.
- Protocol class contains the main method. It deals with instantiating the participants and calling functions for each protocol's phase.

The protocol code was published on GitHub [36] and can be executed in a Google Colab notebook [37].

## 6. Comparison with Other Approaches

Following a similar approach as in [16], we carried out a qualitative comparison between our proposed protocol and the following proof-based consensus protocols: Pure PoW [38], Cuckoo hash function-based PoW [39], Prime number finding-based PoW [23], Double puzzles-based PoW [40], Non-outsourceable puzzles [41], Bitcoin-NG [42], GHOST strategy [43], Generalized PoW [44], Pure PoS (Nextcoin) [45], State of the block-based PoS [19], PoS by coin flipping from many nodes [46], Delegated PoS [47], Coin age-based PoW difficulty re-designation (Ppcoin) [20], Stake-based PoW difficulty re-designation (Blackcoin) [21], Coin age with an exponential decay function [48], Combining PoW and PoS to append blocks sequentially [49], with difficulty adjustment [50], Proof of activity [51], Puzzles designed for human PoW [22], Proof of burn [24], Proof of space [25], Proof of elapsed time [52], Proof of luck [53], Multichain [54]; and the following Vote-based consensus protocols: Hyperledger with practical Byzantine fault tolerance [55], Symbiont, R3 Corda with BFT-SMaRt [56,57], Iroha with Sumeragi [58,59], Ripple [60], Stellar [61], Quorum with Raft [62], Chain [63].

To perform our qualitative comparison, we defined a set of features that all of these protocols shared. Additionally, for each feature, we defined a rank of values with their

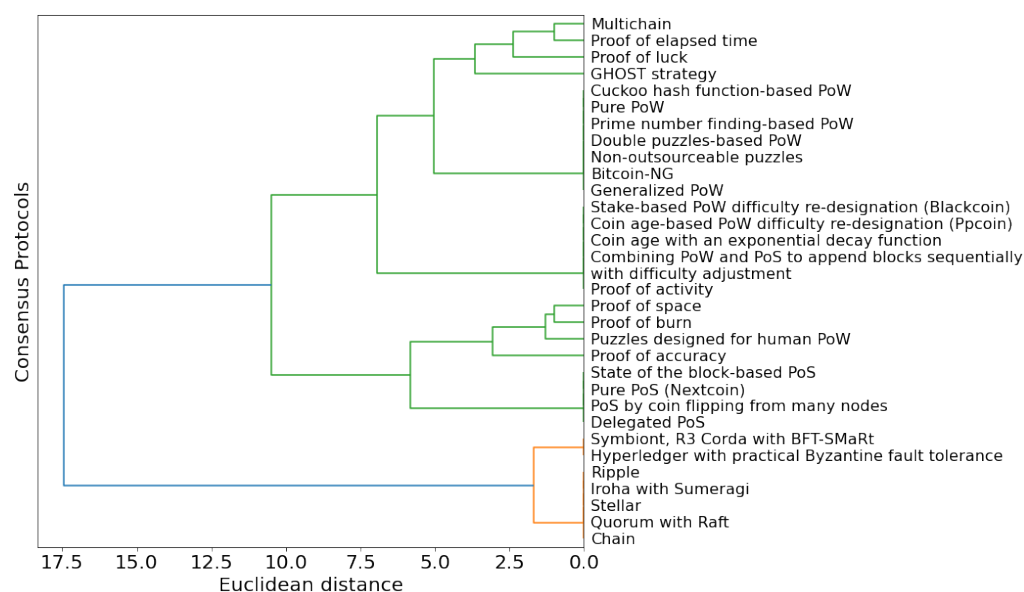
corresponding numeric values. Table 2 shows the defined features with the values they can assume.

**Table 2.** Features and values used to cluster.

Feature	Description	Value
Energy efficiency	Efficient energy used to perform the tasks of the protocol	Yes = 0
		No = 1
Modern hardware	Modern hardware requirements	No need = 0
		Low need = 1
		Need = 2
		High = 3
Forking	Possibility to perform a forking of the main chain	Never = 0
		Very difficult = 1
		Difficult = 2
		Probably = 3 Very Probably = 4
Double-spending attack	Possibility of a double-spending attack	Never = 0
		Difficult = 1
		More or less = 2 Easy = 3
Block creation speed	Speed to create a block	Very fast = 0
		Fast = 1
		Low = 2
		Very low = 3
Mining Pool	Amenable to mining pool creation	Never = 0
		Very difficult = 1
		Can be prevented = 2
		Difficult to prevent = 3 It occurs = 4
Number of participants	Number of participants in the protocol	Mostly unlimited = 0
		Limited = 1
Decentralization	Decentralization of participants	Mostly high = 0
		Low = 1
Trust	Trust of the network	More trustful = 0
		Less trustful = 1
Node Identities	Node identity management	No = 0
		Yes = 1
Security threat	Security threat to network	More serious = 0
		Less serious = 1
Award-giving	Award-giving to miner nodes	Yes = 0
		Mostly no = 1

Based on what is found in the literature, we assigned a value per feature per protocol, creating a dataset whose rows represent the names of protocols and columns represent the features. We then applied hierarchical clustering over the dataset to group the protocols; we worked with the method of least variance (Ward's method); which seeks to obtain the least variability intra-cluster to ensure that each group is the most homogeneous possible.

We finally identified the group (and, hence, the features) in which the proposed protocol (PoAc) was located. Figure 7 shows the groups created by the clustering algorithm.



**Figure 7.** Hierarchical clustering dendrogram.

As a result of the hierarchical clustering, we identified that PoAc is part of a group made up of proof of space, proof of burn, and puzzle designed for human PoW; using as reference the previously defined features, we can say that these protocols share the following features:

- Non-energy-efficient.
- Their need for modern hardware for its execution is low.
- They may present forking.
- Resistant to double-spending attack.
- Strategies may be deployed to prevent mining pools.
- High decentralization.
- Trustful
- Award-giving to a node adding a block to the chain.

We consider that our proposed protocol may be implemented and deployed in any blockchain, which decides the selection of the winning participant (that is, who wins the right to add a new block to the main chain) through a proof of work-based protocol (PoW). Additionally, our proposed protocol does not require the participants to have any particular hardware, making it implementable in blockchains where the hardware of the participants features any computing power.

A particular use case can be a decentralized application for registering anonymous touristic information where each participant in the network corresponds to a touristic operator. For this application, the integrity and availability of the data are crucial, and so there must be a control on registering data in the blockchain (making a consensus necessary). The touristic operators feature pieces of hardware that may not be homogeneous and typically present a low level of computation.

Additionally, this scenario may use its cryptocurrency for the consortium of touristic operators. A tourist can use it to carry out monetary transactions with the different operators, requiring trust between the participants. In addition to the touristic operators and their resources, tourists would participate in the network (even in the protocol) with mobile devices featuring varying computing power.

## 7. Conclusions and Future Work

Although the study of alternative consensus protocols has taken an interest in the scientific blockchain community recently, most of the related works are still theoretical ideas. Proof-of-accuracy protocols are a particular case since there are few papers about them in the literature, presenting neither concrete proof-of-accuracy protocols nor implementations.

This paper introduced a proof-of-accuracy protocol, starting with an initial and insecure design, which we progressively improved regarding security. Our proposed protocol removed the need for a coordinator and combined the proof of work component with access to random locations to improve the protocol's resistance to majority attacks. Our analysis pointed out that it is secure and fair assuming a semi-honest adversary.

In future work, we would like to analyze and improve our proposed protocol in a scenario with dishonest participants, where they may not follow its rules or carry out attacks that may compromise its secret information. Another research topic is to make the protocol quantum-resistant since it is a Diffie-Hellman-like cryptographic construction, which is not quantum-resistant.

**Author Contributions:** Conceptualization, F.A.A.-N. and R.V.-P.; methodology, F.A.A.-N. and R.V.-P.; software, F.A.A.-N. and R.V.-P.; validation, F.A.A.-N. and R.V.-P.; formal analysis, F.A.A.-N. and R.V.-P.; investigation, F.A.A.-N. and R.V.-P.; resources, F.A.A.-N. and R.V.-P.; data curation, F.A.A.-N. and R.V.-P.; writing—original draft preparation, F.A.A.-N. and R.V.-P.; writing—review and editing, F.A.A.-N. and R.V.-P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was carried out with the support of the program “Convocatoria 779 de 2017-Doctorado Nacional Convocatoria para la Formación de Capital Humano de Alto Nivel para el Departamento de Boyacá–2017” and the Universidad del Norte.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

- Pilkington, M. Blockchain Technology: Principles and Applications. In *Research Handbook on Digital Transformations*; Research Handbooks in Business and Management, Chapter 11; Ollerios, F.X., Zhegu, M., Eds.; Edward Elgar Publishing: Cheltenham, UK, 2016; pp. 225–253. [\[CrossRef\]](#)
- Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain Technology: Beyond Bitcoin. *Appl. Innov. Rev.* **2016**, *14*, 5–20.
- IBM. IBM Blockchain: What is Blockchain Technology? 2021. Available online: <https://www.ibm.com/za-en/topics/what-is-blockchain> (accessed on 4 January 2021).
- Zozaya, C.; Incera, J.; Velázquez, A. Blockchain: Un Tutorial. *Estud. Filos. Hist. Let.* **2019**, *17*, 113. [\[CrossRef\]](#)
- Aponte-Novoa, F.A.; Orozco, A.L.S.; Villanueva-Polanco, R.; Wightman, P. The 51% Attack on Blockchains: A Mining Behavior Study. *IEEE Access* **2021**, *9*, 140549–140564. [\[CrossRef\]](#)
- Le, T.V.; Hsu, C.L. A Systematic Literature Review of Blockchain Technology: Security Properties, Applications and Challenges. *J. Internet Technol.* **2021**, *22*, 789–801. [\[CrossRef\]](#)
- Wang, Y.; Kogan, A. Designing Confidentiality-preserving Blockchain-based Transaction Processing Systems. *Int. J. Account. Inf. Syst.* **2018**, *30*, 1–18. [\[CrossRef\]](#)
- Le, T.V.; Hsu, C.L.; Chen, W.X. A Hybrid Blockchain-Based Log Management Scheme With Nonrepudiation for Smart Grids. *IEEE Trans. Ind. Inform.* **2022**, *18*, 5771–5782. [\[CrossRef\]](#)
- Gao, J.; Asamoah, K.O.; Sifah, E.B.; Smahi, A.; Xia, Q.; Xia, H.; Zhang, X.; Dong, G. GridMonitoring: Secured Sovereign Blockchain Based Monitoring on Smart Grid. *IEEE Access* **2018**, *6*, 9917–9925. [\[CrossRef\]](#)
- Javed, I.T.; Alharbi, F.; Bellaj, B.; Margaria, T.; Crespi, N.; Qureshi, K.N. Health-ID: A Blockchain-Based Decentralized Identity Management for Remote Healthcare. *Healthcare* **2021**, *9*, 712. [\[CrossRef\]](#)
- Kurpjuweit, S.; Schmidt, C.G.; Klöckner, M.; Wagner, S.M. Blockchain in Additive Manufacturing and its Impact on Supply Chains. *J. Bus. Logist.* **2021**, *42*, 46–70. [\[CrossRef\]](#)
- Wu, H.; Li, Z.; King, B.; Ben Miled, Z.; Wassick, J.; Tazelaar, J. A Distributed Ledger for Supply Chain Physical Distribution Visibility. *Information* **2017**, *8*, 137. [\[CrossRef\]](#)
- Saberi, S.; Kouhizadeh, M.; Sarkis, J.; Shen, L. Blockchain Technology and its Relationships to Sustainable Supply Chain Management. *Int. J. Prod. Res.* **2019**, *57*, 2117–2135. [\[CrossRef\]](#)



14. Dean. Cryptorials.io. 2015. Available online: <http://cryptorials.io/glossary/51-attack/> (accessed on 1 June 2022).
15. Anita, N.; Vijayalakshmi, M. Blockchain Security Attack: A Brief Survey. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019, Kanpur, India, 6–8 July 2019; pp. 6–11. [CrossRef]
16. Aponte, F.; Gutierrez, L.; Pineda, M.; Meriño, I.; Salazar, A.; Wightman, P. Cluster-Based Classification of Blockchain Consensus Algorithms. *IEEE Lat. Am. Trans.* **2021**, *19*, 688–696. [CrossRef]
17. Oyinloye, D.P.; Teh, J.S.; Jamil, N.; Alawida, M. Blockchain Consensus: An Overview of Alternative Protocols. *Symmetry* **2021**, *13*, 1363. [CrossRef]
18. Kudin, A.M.; Kovalenko, B.A.; Shvidchenko, I.V. Blockchain Technology: Issues of Analysis and Synthesis. *Cybern. Syst. Anal.* **2019**, *55*, 488–495. [CrossRef]
19. Nguyen, G.T.; Kim, K. A survey About Consensus Algorithms Used in Blockchain. *J. Inf. Process. Syst.* **2018**, *14*, 101–128. [CrossRef]
20. Sunny, K.; Scott, N. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Technical Report, Self-Published Paper. 2012. Available online: <https://www.semanticscholar.org/paper/PPCoin%3A-Peer-to-Peer-Crypto-Currency-with-King-Nadal/0db38d32069f3341d34c35085dc009a85ba13c13> (accessed on 1 June 2022).
21. Vasin, P. BlackCoin's Proof-of-Stake Protocol v2 Pavel. 2014. Available online: <https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf> (accessed on 1 April 2022).
22. Blocki, J.; Zhou, H.S. Designing Proof of Human-Work Puzzles for Cryptocurrency and Beyond. In *Theory of Cryptography*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 517–546.
23. Sunny, K. Primecoin: Cryptocurrency With Prime Number Proof-of-Work. 2013. Available online: <http://primecoin.io/bin/primecoin-paper.pdf> (accessed on 1 June 2022).
24. P4Titan. Slimcoin. A Peer-to-Peer Crypto-Currency with Proof-of-Burn “Mining without Powerful Hardware”. Available online: <https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKewiMpdq7rfr4AhUVU3wKHSGoDwUQFnoECAoQAw&url=https%3A%2F%2Fslimcoin.info%2FwhitepaperSLM.pdf&usg=AOvVaw0J3HJ1taDkyMw81CsImBQ1> (accessed on 20 January 2021).
25. Park, S.; Kwon, A.; Fuchsbaauer, G.; Gaži, P.; Alwen, J.; Pietrzak, K. SpaceMint: A Cryptocurrency Based on Proofs of Space. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 480–499.
26. Yu, B.; Liu, J.; Nepal, S.; Yu, J.; Rimba, P. Proof-of-QoS: QoS Based Blockchain Consensus Protocol. *Comput. Secur.* **2019**, *87*, 101580. [CrossRef]
27. Liu, Y.; Liu, J.; Zhang, Z.; Yu, H. A Fair Selection Protocol for Committee-Based Permissionless Blockchains. *Comput. Secur.* **2020**, *91*, 101718. [CrossRef]
28. Saad, M.; Qin, Z.; Ren, K.; Nyang, D.; Mohaisen, D. e-PoS: Making Proof-of-Stake Decentralized and Fair. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1961–1973. [CrossRef]
29. Kaur, M.; Gupta, S.; Kumar, D.; Verma, C.; Neagu, B.C.; Raboaca, M.S. Delegated Proof of Accessibility (DPoAC): A Novel Consensus Protocol for Blockchain Systems. *Mathematics* **2022**, *10*, 2336. [CrossRef]
30. Hoogerwerf, E.; van Tetering, D.; Bay, A.; Erkin, Z. Efficient Joint Random Number Generation for Secure Multi-party Computation. In Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, Lieusant, Paris, 6–8 July 2021; di Vimercati, S., Samarati, P., Eds.; SciTePress: Setúbal, Portugal, 2021; pp. 436–443. [CrossRef]
31. Kursawe, K.; Danezis, G.; Kohlweiss, M. Privacy-Friendly Aggregation for the Smart-Grid. In *Privacy Enhancing Technologies*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6794, pp. 175–191. [CrossRef]
32. Boneh, D.; Shoup, V. *A Graduate Course in Applied Cryptography*; Self-Publishing: Stanford, CA, USA, 2020. Available online: <http://toc.cryptobook.us/> (accessed on 1 April 2022).
33. Evans, D.; Kolesnikov, V.; Rosulek, M. *A Pragmatic Introduction to Secure Multi-Party Computation*; NOW Publishers Inc.: Delft, The Netherlands, 2018; Volume 2. [CrossRef]
34. Ahlgren, J. The Probability Distribution for Draws Until First Success Without Replacement. *arXiv* **2014**, arXiv:1404.1161.
35. Snoeren, M. Python-p2p-Network: Framework to Easily Implement Decentralized Peer-to-Peer Network Applications in Python. 2021. Available online: <https://github.com/macsnoreen/python-p2p-network> (accessed on 10 May 2022).
36. Aponte-Novoa, F.A.; Villanueva-Polanco, R. Proof of Accuracy Consensus Protocol. 2022. Available online: [https://github.com/faan03/proof\\_of\\_accuracy\\_consensus\\_Protocol](https://github.com/faan03/proof_of_accuracy_consensus_Protocol) (accessed on 15 April 2022).
37. Aponte-Novoa, F.A.; Villanueva-Polanco, R. Notebook Proof of Accuracy Consensus Protocol. 2022. Available online: <https://colab.research.google.com/drive/1lIHERlKMckK4vEjzYKqPvvFr7h4SDelZ?usp=sharing> (accessed on 1 June 2022).
38. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *J. Gen. Philos. Sci.* **2008**, *39*, 53–67. [CrossRef]
39. Pagh, R.; Rodler, F.F. Cuckoo Hashing. *BRICS Rep. Ser.* **2001**, *5*. [CrossRef]
40. Eyal, I.; Sirer, E.G. How to Disincentivize Large Bitcoin Mining Pools. Blog Post. 2014. Available online: <http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools> (accessed on 1 April 2022).
41. Miller, A.; Kosba, A.; Katz, J.; Shi, E. Nonoutsourcable Scratch-off Puzzles to Discourage Bitcoin Mining Coalitions. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 680–691.



42. Eyal, I.; Gencer, A.E.; Sirer, E.G.; Van Renesse, R. Bitcoin-NG: A scalable Blockchain Protocol. In Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), Santa Clara, CA, USA, 16–18 March 2016; pp. 45–59.
43. Sompolinsky, Y.; Zohar, A. Accelerating Bitcoin’s Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptol. ePrint Arch.* **2013**, 2013, 881.
44. Tang, S.; Liu, Z.; Chow, S.S.M.; Liu, Z.; Long, Y. Forking-Free Hybrid Consensus with Generalized Proof-of-Activity. *IACR Cryptol. ePrint Arch.* **2017**, 2017, 367.
45. Nxt Whitepaper-Introduction: Nxt Whitepaper. Available online: [https://nxtdocs.jelurida.com/Nxt\\_Whitepaper](https://nxtdocs.jelurida.com/Nxt_Whitepaper) (accessed on 5 June 2022).
46. Ismail, L.; Hameed, H.; AlShamsi, M.; AlHammadi, M.; AlDhanhani, N. Towards a Blockchain Deployment at UAE University: Performance Evaluation and Blockchain Taxonomy. In Proceedings of the 2019 International Conference on Blockchain Technology, ICBCT 2019, Honolulu, HI, USA, 15–18 March 2019. [CrossRef]
47. Proof of Stake vs. Delegated Proof of Stake | Gemini. Available online: <https://www.gemini.com/cryptopedia/proof-of-stake-delegated-pos-dpos#section-delegated-proof-of-stake> (accessed on 5 June 2022).
48. Ren, L. Proof of Stake Velocity: Building the Social Currency of the Digital Age. 2014. Available online: <https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiUuoW2sfr4AhXEU3wKHTNIDvYQFnoECAkQAQ&url=https%3A%2F%2Fcryptochainuni.com%2Fwp-content%2Fuploads%2FReddcoin-Proof-of-Stake-Velocity.pdf&usq=AOvVaw0q9hibSqoHRpsK5rOtlfTv> (accessed on 1 June 2022).
49. Duong, T.; Fan, L.; Katz, J.; Thai, P.; Zhou, H.S. 2-hop Blockchain: Combining Proof-of-Work and Proof-of-Stake Securely. In *Computer Security—ESORICS 2020*; Chen, L., Li, N., Liang, K., Schneider, S., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 697–712.
50. Chepur, A.; Duong, T.; Fan, L.; Zhou, H.S. TwinsCoin: A Cryptocurrency via Proof-of-Work and Proof-of-Stake. *IACR Cryptol. ePrint Arch.* **2018**, 2017, 232.
51. Bentov, I.; Lee, C.T.; Mizrahi, A.; Rosenfeld, M. Proof of Activity: Extending Bitcoin’s Proof of Work via Proof of Stake [Extended Abstract]. *IACR Cryptol. ePrint Arch.* **2014**, 2014, 452. [CrossRef]
52. Sawtooth. Available online: <https://sawtooth.hyperledger.org/docs/1.2/> (accessed on 5 June 2022).
53. Milutinovic, M.; He, W.; Wu, H.; Kanwal, M. Proof of Luck. In Proceedings of the 1st Workshop on System Software for Trusted Execution, Trento, Italy, 12 December 2016. [CrossRef]
54. Greenspan, G. Multichain Private Blockchain-White Paper. 2015, p. 85. Available online: <http://www.multichain.com/download/MultiChain-White-Paper.pdf> (accessed on 1 June 2022).
55. Hyperledger—Open Source Blockchain Technologies. Available online: <https://www.hyperledger.org/> (accessed on 5 June 2022).
56. Symbiont-Enterprise Fintech Using Blockchain Technology. Available online: <https://www.symbiont.io/> (accessed on 5 June 2022).
57. Corda. Available online: <https://www.corda.net/> (accessed on 5 June 2022).
58. Iroha Whitepaper. Available online: [https://github.com/hyperledger/iroha/blob/iroha2-dev/docs/source/iroha\\_2\\_whitepaper.md#28-consensus](https://github.com/hyperledger/iroha/blob/iroha2-dev/docs/source/iroha_2_whitepaper.md#28-consensus) (accessed on 5 June 2022).
59. Hyperledger Iroha Documentation. Available online: <https://iroha.readthedocs.io/en/develop/> (accessed on 5 June 2022).
60. Schwartz, D.; Youngs, N.; Britto, A. *The Ripple Protocol Consensus Algorithm*; Ripple Labs Inc White Paper: San Francisco, CA, USA, 2014; Volume 5, p. 151. Available online: [https://reasonabledeviations.com/notes/papers/ripple\\_consensus\\_protocol/](https://reasonabledeviations.com/notes/papers/ripple_consensus_protocol/) (accessed on 1 June 2022).
61. Mazieres, D. *The Stellar Consensus Protocol: A Federated Model For Internet-Level Consensus*; Stellar Development Foundation: San Francisco, CA, USA, 2015.
62. Lamport, L. *Paxos Made Simple*; ACM SIGACT News (Distributed Computing Column), 2001; Volume 32. Available online: <https://www.microsoft.com/en-us/research/uploads/prod/2016/12/paxos-simple-Copy.pdf> (accessed on 1 June 2022).
63. Federated Consensus. Available online: <https://chain.com/docs/1.2/protocol/papers/federated-consensus/> (accessed on 5 June 2022).