DONS: Dynamic Optimized Neighbor Selection for Smart Blockchain Networks*

Hamza Baniata[a,*], Ahmad Anaqreh[b] and Attila Kertesz[a]

[a]Department of Software Engineering, University of Szeged, Szeged 6720, Hungary
[b]Department of Computational Optimization, University of Szeged, Szeged 6720, Hungary

ARTICLE INFO

Keywords:
Smart Networking
Blockchain
Optimized Neighbor Selection
Minimum Spanning Tree
Anonymized Leader Election

ABSTRACT

Blockchain (BC) systems mainly depend on the consistent state of the Distributed Ledger (DL) at different logical and physical places of the network. The majority of network nodes need to be enforced to use one or both of the following approaches to remain consistent: i) to wait for certain delays (i.e. by requesting a hard puzzle solution as in PoW and PoUW, or to wait for random delays as in PoET, etc.) ii) to propagate shared data through shortest possible paths within the network. The first approach may cause higher energy consumption and/or lower throughput rates if not optimized, and in many cases these features are conventionally fixed. Therefore, it is preferred to enhance the second approach with some optimization. Previous works for this approach have the following drawbacks: they may violate the identity privacy of miners, only locally optimize the Neighbor Selection method (NS), do not consider the dynamicity of the network, or require the nodes to know the precise size of the network at all times. In this paper, we address these issues by proposing a Dynamic and Optimized NS protocol called DONS, using a novel privacy-aware leader election within the public BC called AnoLE, where the leader anonymously solves the The Minimum Spanning Tree problem (MST) of the network in polynomial time. Consequently, miners are informed about the optimum NS according to the current state of network topology. We analytically evaluate the complexity, the security and the privacy of the proposed protocols against state-of-the-art MST solutions for DLs and well known attacks. Additionally, we experimentally show that the proposed protocols outperforms state-of-the-art NS solutions for public BCs. Our evaluation shows that the proposed DONS and AnoLE protocols are secure, private, and acutely outperform all current NS solutions in terms of block finality and fidelity.

1. Introduction

Blockchain (BC) [1] is the backbone of the famous, robust and reliable P2P Bitcoin network, which proposes many simple solutions for different problems that faced a successful distributed digital currency system for years. One of those problems was the consistency of the Distributed Ledger (DL) at any given time [2]. A system is consistent, when it ensures that every reading is the same on any node, i.e., the nodes have a global view of the system state [3, 4]. Different criteria imposes different readings, e.g. the fluctuating transmission delay between nodes and the continuous alteration of data [5]. Although BC did not directly solve this open problem, it founded an approach assuring data saved on the DL would be synchronized soon enough, so that the DL is consistent [6]. Previous studies show that in a BC-based DL, more neighbors per miner and higher delivery time rates between neighbors, both lead to lower levels of DL consistency [7, 8]. These results served as a motivation to our research for designing an optimized BC networking and gossiping protocol. Such a protocol shall require minimum number of neighbors per miner, directing the miners to communicate with globally-optimized selection of neighbors.

A scalable system is one that maintains constant, or slowly degrading, overheads and performance as its size increases [9]. The dynamicity in P2P networks, which are the physical infrastructure of BCs, imposes even more complicated problems than the DL inconsistency as it directly affects its scalability. That is, the constantly changing topology of the network leads to non-consistent propagation delays between its entities. BC peers are connected to several neighboring peers, and adopt a Random Neighbor Selection (RNS) with which they share data [10]. Generally, shared data between peers include new blocks or information regarding the state of the sender's ledger (i.e. gossiping). Gossiping also includes sharing the best DL version between peers, which is defined according to certain criteria (e.g. longest chain). The RNS method implies randomized paths, walked by shared data [11], leading to an inefficient data propagation scheme. This is due to several redundant exchanged messages caused by the probability of cycle appearance on the randomly selected path of data, leading to higher average finality time and lower consistency levels.

Although it is not an optimized method, RNS is currently adopted by most BC systems. Those BCs compensate the high finality time by enforcing miners to spend more time solving the puzzle. This compensation technique does indeed achieve its goal, yet it leads to both lower overall sys-

*This research work was supported by the Hungarian Scientific Research Fund under the grant number OTKA FK 131793, and by the TruBlo project of the European Union's Horizon 2020 research and innovation program under grant agreement No. 957228, and by the National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme, and by the University of Szeged Open Access Fund under the grant number 5544.

*Corresponding author
baniatah@inf.u-szeged.hu (H. Baniata)
ORCID(s): 0000-0001-7511-2910 (H. Baniata); 0000-0002-3971-2684 (A. Anaqreh); 0000-0002-9457-2928 (A. Kertesz)

tem throughput (due to delayed new block generation), and higher energy consumption (in case the puzzle solution consumes energy. e.g. the PoW and the PoUW consensus models). Few other methods were proposed to locally optimize Neighbor Selection (NS), and indeed addressed the dynamicity issue (e.g. [12]). However, none of these solutions proposed *optimum* NS.

We envision a better solution to allow peers to communicate with selected neighbors according to globally optimized criterion. This criterion has to fulfil mainly three conditions:

1. It decreases the number of cycles within a path, that shared data walks, from *any* peer to *any* other peer (i.e. no cycles, hence a Spanning Tree is an optimal solution [13]).
2. It decreases the maximum time spent from generating data, by any peer, till it reaches all the peers of the network.
3. It addresses the scalability issues of the network, leading to *adaptive* optimization of NS *in spite of* continuous change in network topology.

The optimum selection of paths within a connected network, such as the discussed P2P BC network, is in fact finding the Minimum Spanning Tree (MST) of the network. Utilizing the MST of a given BC network shall lead to increased number of peers receiving a shared piece of data in minimum time, which results in both enhanced data finality and enhanced DL consistency.

In this paper, we propose a Dynamic and Optimized Neighbor Selection protocol called DONS that computes the MST of a public BC network, while preserving the privacy of the participating peers. DONS is also able to dynamically update the MST as peers join and leave the network. This protocol includes a privacy-preserving leader election method, allowing one of the peers within the BC network to compute the MST without previous knowledge of network peers identities (e.g. IP address). The leader nominates itself and becomes active once the majority of voters accepted it as a leader within a predefined round-time. Once active, the leader builds a global demonstration of the BC network topology. The local views sent by voters contain no private data of the senders and, thus, these views can only be used to determine an anonymized topology of the network. Using one of the famous MST algorithms (e.g. Prim's or Kruskal's), the leader computes the MST and broadcasts it to all the network. Every recipient of the MST then can read only its identity and its neighbors' identities, leading to each peer of the network communicating with the optimized selection of neighbors. As a result, our current research work addresses the Neighbor Selection Problem (NSP) of public BCs.

Although the recipient peers can then know the anonymized MST, they cannot, by any means, know the identities of peers other than themselves and their neighbors. We evaluate the proposed DONS protocol against other approaches, utilizing two randomized network models, namely Erdős-Rényi (ER) model [14] and Barabási-Albert (BA) model [15]. The DONS protocol is analytically evaluated in terms of security and privacy [16], and is experimentally evaluated in terms of propagation time and message overhead against the currently used RNS and local RTT-based NS methods. The leader election method is theoretically and experimentally evaluated, in terms of time and message complexity [17], against a recent solution proposed in [18]. The results of our evaluation shows that our proposed protocols are secure, private, efficient and significantly outperform the current related methods.

As will be discussed in later sections, we found no previous work that specifically proposed a privacy-aware leader election method, within the frame of public permissionless BCs, and deployed it to dynamically solve the NSP by finding the network MST. To the best of our knowledge, this is the first research paper that proposes such a protocol.

The remainder of this paper is organized as follows: Section 2 discusses the state-of-the-art regarding the NSP of public BC networks, MSTP, and the leader election problem. Section 3 defines the basic preliminaries and notations on which we build our methods. Section 4 presents the proposed DONS and AnoLE protocols. Section 5 presents our evaluation of the proposed protocols in terms of privacy, security, time and message complexity, finality and fidelity. Section 6 discusses the proposed protocol in terms of future potential and open issues. Finally, Section 7 concludes our work.

2. Related work

Finding the MST of general distributed systems by different means, for example by building a binary tree in distributed fashion within the network and select the root node to search for shortest paths, have already been proposed in the literature [19, 20]. Additionally, many leader election algorithms have been proposed within other contexts, e.g. general distributed systems, or even BC networks, that do not consider the identity privacy as a constraint [21], or ad-hoc wireless sensor networks that have gateway controllers [22, 23]. In those methods, the leader is utilized to administrate the network, mine new blocks, select next miners, or perform specific computations for specific slave nodes [24, 9].

2.1. NSP in public BC networks

It has been shown in several previous works how optimizing the NS decreases the probability of DL forking [25]. In this subsection, we investigate approaches other than RNS [26], as it is the most used in BC networks while it is the least optimized. Examples of such networks include Bitcoin [1] and Hyberledger Fabric [27].

Bi et al. [12] proposed a latency-based NS protocol where miners measure the Round Trip Time (RTT) to their neighbors. Accordingly, miners favor neighbors with lowest RTT, when they need to perform NS. Similarly, a bandwidth informed NS protocol was proposed by Wang and Kim [28], where BC miners favor communications with neighbors that offer higher bandwidth transmission. Accordingly, more data

may flow through the network as links with relatively limited bandwidth are ignored, leading to decreased congestion and enhanced overall throughput. Aoki and Shudo [29] proposed a score-based NS protocol where each miner scores its neighbors according to difference between block generation time (which is typically consisted as a timestamp within a shared block), and block receiving time at the receiver side. That is, a neighbor that usually delivers new blocks faster than other neighbors shall have better communications with the network, thus is given higher score. Consequently, miners favor neighbors with higher scores when they need to perform NS. Notably, this method suggests that miners select neighbors to communicate with depending on the history of the neighbors, which implies that it may take much time to arrive to optimal NS in dense and dynamic networks.

Jin et al. [30] proposed clustering the Bitcoin network and sharing the Transaction (TX) IDs instead of the TXs themselves. That is, TXs to be shared with neighboring clusters are only shared with them if they have not been yet received. Exchanged messages are, then, shared with targeted destinations rather than in a randomized fashion. The proposal was found efficient in terms of network traffic, yet it deviates the network model from distributed towards centralized, as each cluster has its own leader. Additionally, security and privacy analysis were not conducted, although the leader election protocol, performed within each cluster, required private information to be shared among the cluster (e.g. number of Bitcoins obtained by each miner, length of online time, miners' IDs, etc.).

Yu et al. [31] proposed that a shared block within a BC network shall include the IDs of miners who have already received it. Each recipient adds its ID to each received block, and forwards it to all neighbors who haven't yet received it. Apparently, such approach requires a tree topology of the network, which is not guaranteed in public BCs, and implies that shared blocks are constantly modified. To solve the first issue, a method to divide the network into subareas was proposed. The second issue, however, may raise concerns regarding the credibility of shared data as some nodes may behave dishonestly. Similar approaches were proposed in Li [32] and He et al. [33], where multi-link concurrent communication schemes were utilized. The adoption of tree structures was recommended so that a failure node shall only isolate a sub-tree compared to a whole component in case the network topology was mesh. Obviously, such proposal includes several conditions that do not necessarily apply in current public BCs. On the other hand, He et al. [33] proposed that each miner maintains a locally saved historical log of peers' IDs. Referring to this log, miners may select peers to gossip with if they are not in the historical log.

To summarize our survey, all of the presented approaches indeed perform better than the currently adopted RNS approach. However, all of these approaches address the NSP depending on local views of the network, leading to local NS optimization. Additionally, a group of those approaches requires modifying the underlying network topology and/or violates the identity privacy constraints usually present in public BCs.

We argue that a protocol that solves the NSP can be assumed comprehensive if it fulfils three main criteria: 1) It optimizes the NS depending on a *global* view of the network topology in a timely manner 2) It requires no modification of the underlying network topology and 3) It preserves the *Identity privacy* of all peers within the network. As none of the current protocols has fulfilled these requirements, one can state that the NSP has not yet been solved for public BCs.

2.2. Semi-Distributed Minimum Spanning Tree (Semi-DMST)

To fulfil the first criterion of a comprehensive protocol that addresses the NSP, one needs to utilize the global view of the BC network using Graph Theory. Using this representation, one can notice that solving its NSP is a central optimization problem, namely the Minimum Spanning Tree Problem (MSTP) [13]. That is, finding the MST of the graph that represents the BC network is, in fact, finding the global solution of the NSP. This approach also fulfils the second criterion above, as no new edges are enforced into the graph.

It is trivial to find the MST of a given network in polynomial time, if its topology is known, using famous algorithms such as Prim's [34] or Kruskal's [35]. Accordingly, BC networks that consist of a Trusted Third Party (TTP, which tracks system entities and is trusted to build a global view relation graphs demonstrating the network) can calculate the MST using one of the well known algorithms. However, public and permissionless BCs don't usually consist of a TTP, which implies that no entity within the network can build a graph that demonstrates the network. Accordingly, Prim's, Kruskal's, or any other algorithm that requires a global view of the network, cannot be used in fully distributed BCs.

Since fully-distributed BC networks are actually distributed systems, the NSP in those BC networks can be formalized using the Distributed Minimum Spanning Tree (DMST) problem [36]. This problem aims at computing the MST of a distributed system without prior knowledge of network topology. This problem has a long line of research dating back to 1926 [37], until 2018 when the problem could finally enjoy a singular optimality state with the protocol proposed by Pandurangan et al. [18]. That is, the proposed algorithm solved the DMST problem with, simultaneously, optimal time and optimal message complexity.

Although DMST problem has been solved in [18], and can theoretically be deployed in public BCs, it actually cannot be adopted by current public BCs. That is, the algorithm requires its participants to share their identities, along with other (perhaps considered private) data. Such requirement imposes a privacy issue that will mostly forbid public BCs from utilizing the solution of [18].

According to this brief description of MST and DMST problems and their solutions, the former can be easily utilized in any BC that consists a TTP, a network administrator who has a global view of the network, or a gateway through which new miners shall be confirmed. Specifically, the TTP

periodically finds the MST of the network using one of the previously mentioned methods. Accordingly, the TTP suggests to miners the Optimum Neighbour Selection (ONS) to enhance overall system efficiency. As the condition of a TTP does not apply to public permissionless BCs, such approach does not fulfil the first criterion of a comprehensive NS solution.

The DMST problem, and its solution, can be used to optimize NS in public-permissionless BCs as long as peers trust all other peers with their private identities. That is, each node can deduce the ONS according to data aggregated to it from all nodes of the network, and it can communicate with all nodes in return. However, such approach does not fulfil the third criterion of a comprehensive NS solution.

Following these analysis, this paper attempts to solve Semi-Distributed MST problem, which formalizes the NSP in public-permissionless BCs. In such problem model, peers do not trust each other with their private IDs (except for their neighboring peers), while the network is dynamic and does not have a TTP. The solution we propose for this problem is detailed in Section 4.

2.3. Leader election problem

A leader node in a distributed system might be needed to perform one or more centralized tasks. In static networks, a leader node might be statically configured, or periodically re-selected according to a predefined criteria. Leaders are, then, similar in their properties to all other nodes. Selecting a leader node in dynamic networks, however, is a well known problem for distributed, P2P systems, namely the Leader Election Problem [38]. Depending on the various aspects of the studied distributed system, such as network topology, type of nodes, system architecture, communication channels etc., many leader election algorithms have been proposed. Examples of such solutions include Abraham et al. [39], Al Refai [40, 41] and Biswas et al. [42].

Mapping the Leader Election Problem to a public and permissionless BC, the number of nodes and the upper limit of nodes cannot be specified. Such problem projection requires a solution with more restrictions and higher levels of uncertainty. Nevertheless, previously proposed solutions addressed such challenging settings even with mobile distributed systems and wireless communications (which is not the case in the vast majority of BC-based networks, yet even if it was, it is solved). Examples for single leader election algorithms that can be deployed in a public permissionless BC include the TORA algorithm [43], Malpani algorithm [44] and SEFA algorithm [45]. Examples for multi-leader election algorithms proper for such BC model settings include Kelea [46] and [47]. Nonetheless, those algorithms are of general usability and were not specifically proposed for BC systems. The most famous leader election method, specifically implemented for different BC network models is the RAFT election [48]. RAFT is, basically, one step among many in the RAFT consensus protocol. This step can be utilized within different scenarios. The RAFT leader election process fulfills the three main requirements of a successful leader election, namely safety, liveness and fairness [49].

Overall, most of these leader election algorithms are implementable in public BCs, if sharing the IDs and domains (e.g. IP-addresses) of network peers have no privacy implications. That is, if peers of the network trust *all* other peers with their identity information. However, this is generally not the case in public BCs, where each peer is only aware of its neighbors' identities.

For such requirements, a deterministic and privacy-aware leader election protocol, namely the Right-of-Stake (RoS), was recently proposed by Tan et al. [50]. The RoS protocol suggests that a peer elects itself according to local information, namely its stake and its counter. Accordingly, if the peer fulfilled a given condition, it starts behaving as a leader. Consequently, other peers receiving data from the elected peer confirm it is sent by a leader once the leader reveals its ID. Thus, only when the leader is elected, it will give up its private ID and stake value to other peers. Although this is indeed a privacy aware leader election suitable for public BCs, the authors assumed that the BC network is distributed among pools for necessity. Additionally, the RoS protocol is suitable for synchronous BC networks. Those two conditions limit the utilization of this leader election protocol in case of asynchronous networks or non-pooled BCs. Note that nodes of the network shall be eventually aware of the real identity of the leader, which violates the third criterion of a comprehensive protocol that solves the NSP in public-permissionless BCs. This issue was addressed in [51], yet the proposed algorithms assumed that network nodes are initially aware of the network size. Such information is not necessarily available for miner nodes of public- permessionless BCs, thus, the proposed algorithms are irrelevant for our application.

According to the presented literature review of leader election protocols, we found no previous work that is applicable to the problem of our current research. Thus, we propose a novel protocol, namely AnoLE to address the leader election requirements we seek. We formally define our research problem in Section 3 and detail the proposed solutions in Section 4.

3. Preliminaries and problem statement

Referring to [52], we define a public-permissionless BC network as a connected, undirected, and weighted graph $G = (V, E, w)$, where V is the set of nodes in G representing miner nodes, E is the set of edges in G, representing the communication lines between the miners, where each $e_{i,j} \in E$, connecting exactly two nodes $i, j \in V$, can be traveled in both directions. The nodes of G communicate by message passing via (strictly) the edges of G. Each $e \in E$ is associated with a distinct non-negative value, namely weight ($w_{i,j}$ or w_e), which represents the transmission time needed to deliver 1 bit of data from node i to node j or vice versa, computed in μs.

The weight of any given graph is the sum of the weights of all its edges. We define the set of neighbors of a node $i \in V$ as $m_i = (m_{i,1}, ..m_{i,j})$. We assume that every node

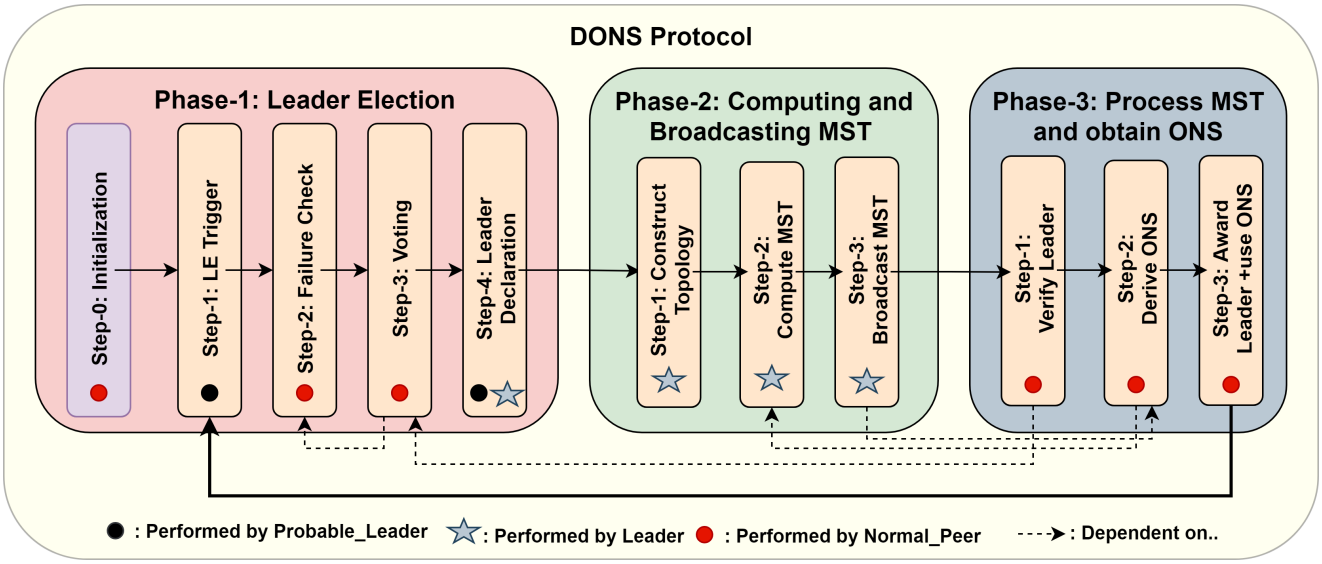

Figure 1: Phases and steps of the proposed DONS protocol. Each step is performed by one (or more) system entity(s). A step may depend on the result of a preceding step of the current round, or on the result of a subsequent step of the previous round

$i \in G$ is initially aware of its m_i, and is aware of the weight associated with each edge $e_{i,j}$ connecting it with any of its neighbors. To mathematically represent a graph, we use the adjacency matrix, which is a matrix of size |V × V|. The elements of the matrix are the weights $w_{i,j}$ if there is $e_{i,j}$ and the maximum size of an integer provided by the interpreter otherwise.

A sub-graph of G' is any graph $G'(V', E', w')$, such that $V' \subseteq V$ and $E' \subseteq E$. G' is also connected, undirected, and weighted as it inherits the properties of the original graph. A Spanning Tree (ST) of G is a connected acyclic sub-graph of G where $V' = V$ and $E' = V - 1$. A Minimum Spanning Tree (MST) of G (with distinct w_e $\forall e \in E'$) is a unique ST where the weight of MST is minimum compared to all STs of G.

A hashing function, or a one-way encryption function, $h(.)$ is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence that is computationally difficult to invert [53]. A hashing function enables the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest[54].

Our research problem is to find the subset $k_i = (k_1, ..k_n) \in m_i$, $\forall i \in V$, such that $e_{i,k} \in MST_G$ $\forall k \in k_i$. We call the solution of our problem the Optimum Neighbor Selection of node $i \in V$ (ONS_i) of a public-permissionless BC network. We aim at solving this problem using a protocol that fulfills the following privacy condition:

$$\forall i \in V \Rightarrow \sigma_i' = \sigma_i + ONS_i \quad (1)$$

where σ_i is the total knowledge of miner i before starting the protocol and σ_i' is the total knowledge of miner i after the protocol is terminated.

4. The Proposed DONS Protocol

In this section, we describe the phases of the proposed DONS protocol and the proposed algorithms and methods for each phase. The generalized framework addresses a public permissionless BC with no TTP, and initially assumes all network entities are honest. However, we discuss counter assumptions where applicable. The phases and steps of the DONS protocol are demonstrated in Figure 1.

4.1. Phase-1: Leader Election

First of all, the DONS protocol requires a global view of the underlying BC network, so that the MST can be computed. Additionally, miners joining and leaving the network implies that this global view, and accordingly the computed MST, shall be regularly updated. In a public and permissionless BC model, all BC miners have the same access permissions and the same level of abstraction. However, one (or a committee) of these miners may perform the MST computations for all other miners. This way, the network decides best practices regarding networking and gossiping without administrative interference, which leads to Smart Networking [55].

In this section, we propose the Anonymous Leader Election (AnoLE) protocol which shall not violate any of the comprehensive NS solution criteria discussed in Section 2.1. The elected leader (single leader in our current work) shall collect non-private local views from all peers, construct a network demonstration, solve the MST problem of the network graph, and finally broadcast the anonymized MST throughout the network. The recipient nodes shall only be able to read their own, and their neighbors' IDs. Thus, neither the leader nor any other network entity can deduce miners' private data throughout the run of the protocol. Note that this condition implies that a miner does not know the identity of the leader, unless the miner itself (or one of its neighbors)

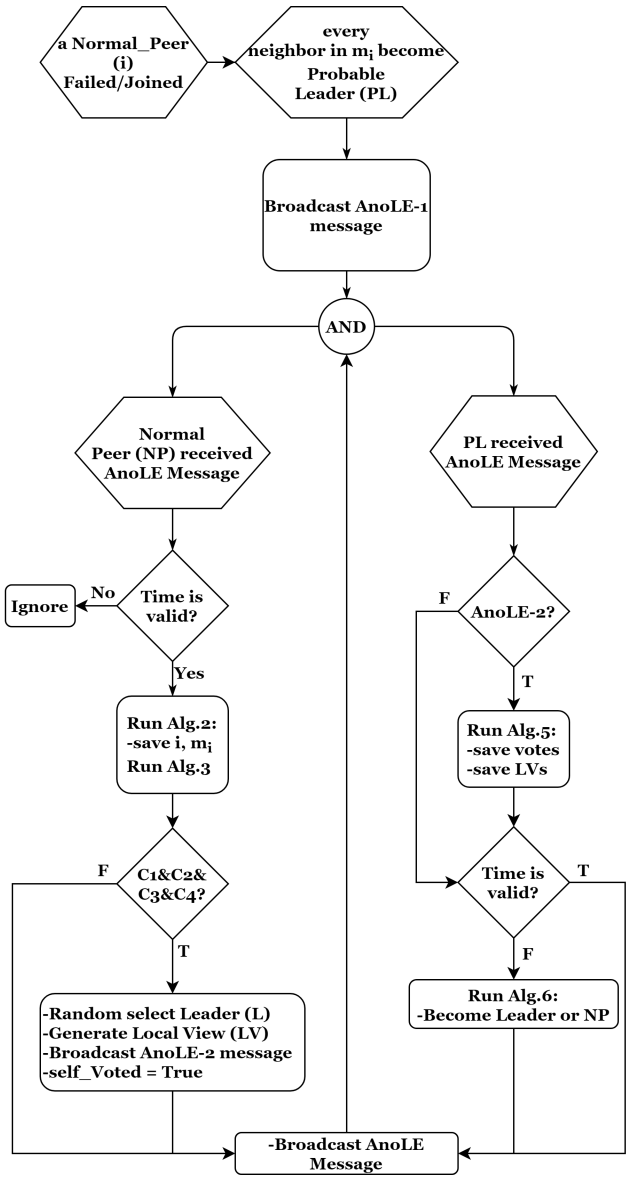

Figure 2: Workflow of the proposed Anonymous Leader Election (AnoLE) protocol

was the leader.

The challenge of this phase is to dynamically select the leader(s). Specifically, selected leader(s) has similar properties to all other nodes, such as failure/unavailability probability (with different failure rates), and nodes being aware of minimum propagation delays only with their adjacent neighbors. The generalized workflow of the proposed AnoLE protocol is depicted in Figure 2. Different system entities utilize the AnoLE protocol as follows:

- Step-0 (Initialization): All nodes know their neighbors identities and the corresponding Round-Trip-Time (RTT) expected when communicating with each of them. All nodes use this protocol honestly, with default status 'Normal_Peer', Default Required Confirmations (DRCs) equals the average number of neighbors per peer, 'Current_Leader' = null, default round time T, and MST set to empty array.
- Step-1 (LE trigger): Once a node i fails/joins the network, its neighbors, denoted $m_i = \{m_{i,1}, ..m_{i,j}\}$, are triggered to start the AnoLE protocol. Each neighbor $m_{i,k} \forall k \in 1, ..j$ sets its status to 'Probable_Leader', sleeps for an arbitrary time (default setting: waiting time is randomly selected between 0 and $T/2$), and sends 'AnoLE-1' message to all neighbors. The 'AnoLE-1' message contains $h(i)$ and $h(m_{i,k})$, along with timestamp t. votes = Dict{} and LVs = list[] are initiated to later save the responses of the AnoLE-1 message.
- Step-2 (Failure check): all nodes that receive 'AnoLE' messages, run Algorithm1. Accordingly, Normal_Peers (NPs) run Algorithms 2, 3 and 4, sequentially, in order to obtain the Required Confirmations (RCs), a list of hashes of nodes in the set m_i, and the Local View (LV), respectively. These recipient nodes wait until they receive a number of distinct AnoLE messages equal to RCs (if T time units passed with no sufficient AnoLE messages, the node does not vote). The recipient checks three conditions to consider the node failure/joining reports correct: 1) Every distinct AnoLE message shall contain similar $h(i)$ and different $h(m_{i,k})$. These messages represent failure/joining proofs 2) Each $h(m_{i,k})$ should belong to the list of Neighbors returned by Algorithm 3, which assures that reports are only sent by genuine neighbors and 3) All neighbors in this list shall send an AnoLE-1 message. This represents a consensus among neighbors on the honesty of the AnoLE protocol trigger (i.e. those neighbors do not know or trust each other by assumption).

  In the special case of an empty MST (which happens only at the first time the protocol is run), recipients ignore conditions 2 and 3. When i is joining the network, the DRCs is used instead of the RCs.
- Step-3 (Voting): Once a recipient node receives a sufficient number of 'AnoLE' messages that fulfill the conditions in Step-2, the recipient can be sure that i has indeed failed/joined as all its neighbors witnessed. The NP then selects one of the received $h(m_{i,k})$s according to a predefined criteria (e.g. randomized, first sender, highest hash value, etc.) and modifies its 'Current_Leader' to the selected $h(m_{i,k})$. After that, NPs broadcast 'AnoLE-2' messages to all its neighbors, which contain their hashed IDs along with the contents of 'AnoLE-1'. AnoLE-2 messages, then, declare that NPs who generated them vote for, specifically, the candidate leader whose hash is included in their AnoLE-2 message. In the context of the DONS protocol, NPs also deposit their current LVs of the network into their generated AnoLE-2 messages. LV is obtained by running Algorithm 4. NPs who have a previous version of the MST (i.e. obtained from previous AnoLE protocol run), may utilize it to share their AnoLE messages with their ONSs. A condition to be fulfilled in order

to utilize the previous ONS, however, is that non of the ONS members has an ID whose hash is equal to $h(i)$.

- Step-4 (Leader Declaration): Whenever a message of type 'AnoLE-2' is received by a 'Probable_Leader', it runs Algorithm 5, which saves new votes and LVs. Once a 'Probable_Leader' finds that: current_time - t $\geq T$, it runs Algorithm 6, which counts the votes received so far and converts the node's status into either 'Leader' or 'Normal_Peer'.

Algorithm 1: Message handler

```
Input AnoLE_msg;
Function Share_msg_with_neighbors(msg)
    if ONS and AnoLE_msg[h(i)] not in ONS then
        neighbors = ONS
    else
        neighbors = self.neighbors
    end
    for neighbor in neighbors do
        send(msg, neighbor)
    end
end
if AnoLE_msg['type'] == 'AnoLE-3' and
  current_leader == AnoLE_msg['leader'] then
    run Algorithm 8
else
    if self.status == 'Normal_Peer' then
        if current_time - AnoLE_msg[t] < T then
            run Algorithm 2;
            Share_msg_with_neighbors(AnoLE_msg)
        end
    end
    if self.status == 'Probable_Leader' then
        if current_time - AnoLE_msg[t] ≤ T then
            Share_msg_with_neighbors(AnoLE_msg)
            if AnoLE_msg.type == 2 then
                run Algorithm 5
            end
        else
            run Algorithm 6
        end
    end
end
```

The identification criteria of nodes can be selected upon system design. That is, in a public permissionless BC, such as Bitcoin, pseudonyms are used to preserve the privacy of end-users [1]. However, true identities in a private or permissioned BC may be used. We believe the distinction between identity management schemes, and thus the adoption of one over the other, is beyond the scope of our work. That is, the selection of an identity management scheme is dependant on/related to the application definition and the required

Algorithm 2: Check local records

```
Input AnoLE_msg;
if AnoLE_msg[h(i)] in self.AnoLE_records then
    if AnoLE_msg[h(m_{i,k})] NOT in
      self.AnoLE_records[h(i)]['Ks'] then
        self.AnoLE_records[h(i)]['Ks'].append(h(m_{i,k}))
    end
else
    self.AnoLE_records[h(i)] = Dict{}
    self.AnoLE_records[h(i)]['Ks'] = h(m_{i,k})
    self.AnoLE_records[h(i)]['voted'] = False
end
M_K, RC = RFC(AnoLE_msg) (Algorithm 3)
if M_K is NOT empty then
    C_1 = AnoLE_msg[h(m_{i,k})] in M_K
    C_2 = M_K ∈ self.AnoLE_records[h(i)]['Ks']
else
    C_1 = C_2 = True
end
C_3 = len(self.AnoLE_records[h(i)]['Ks']) ≥ RC
C_4 = NOT self.AnoLE_records[h(i)]['voted']
if C_1 AND C_2 AND C_3 AND C_4 then
    self.Current_Leader =
      self.AnoLE_records[h(i)]['Ks'][0]
    my_LV = LV_Computation() (Algorithm 4)
    my_AnoLE-2 = [AnoLE_msg[t], h(self.ID),
      AnoLE_msg[h(i)], self.Current_Leader,
      my_LV]
    self.AnoLE_records[h(i)]['voted'] = True
    Share_msg_with_neighbors(my_AnoLE-2)
end
```

Algorithm 3: RFC

```
Input AnoLE[h(i), h(m_{i,k}), t];
Function FIND_neighbors(entity)
    Neighbors = List[]
    for row, column in MST do
        if row[0] == entity and MST[row][column]
          < infinity then
            Neighbors.append(MST[0][column])
        end
    end
    return Neighbors
end
list_of_m = List[];
if MST is NOT empty then
    list_of_m = FIND_neighbors(h(i))
end
RC = len(list_of_m);
if RC == 0 then
    RC = DRC
end
return list_of_m, RC
```

Algorithm 4: Local View (LV_i) *Computation*

```
Anonymized_LV = List[];
Hashed_IDs = List[];
Weights = List[];
for k in m_i do
    Hashed_IDs.append(h(k));
    Weights.append(RTT(k)/2)
end
Anonymized_LV.append(Hashed_IDs);
Anonymized_LV.append(Weights);
return Anonymized_LV
```

Algorithm 5: Voting

```
Input AnoLE-2[t, h(voter), h(i), h(m_{i,k}), LV];
if NOT votes[h(i)] then
    votes[h(i)] = Dict{timestamp,
    info:{'voters': [h(voter)],
    'votes': 1}}
    self.LVs.append([h(i), h(voter), AnoLE-2[LV]])
end
if NOT votes[h(i)][h(m_{i,k})] then
    votes[h(i)]['info'][h(m_{i,k})] = Dict{'voters':
      [h(voter)];'votes': 1}
    self.LVs.append([h(i), voter, AnoLE-2[LV]])
end
if NOT h(voter) in
  votes[h(i)]['info'][h(m_{i,k})]['voters'] then
    votes[h(i)]['info'][h(m_{i,k})]['voters'].append(h(voter))
    votes[h(i)]['info'][h(m_{i,k})]['votes'] += 1
    self.LVs.append([h(i), h(voter), AnoLE-2[LV]])
end
```

Algorithm 6: Leader Recognition

```
Input AnoLE[t, h(i)];
global_max_votes = 0;
Leader = null;
for PL in votes[h(i)] do
    PL_votes = PL['votes']
    if PL_votes > max_votes then
        max_votes = PL_votes
        Leader = PL
    end
end
self.current_leader = PL
if Leader == h(self.ID) then
    self.status = 'Leader'
else
    self.status = 'Normal_Peer'
end
```

trust model. Our proposal, on the other hand, is specifically concerned with the optimization of neighbor selection, which shall be related to both the Consensus and the Network layers of any given BC. Consequently, no matter what identity scheme is applied, our proposed AnoLE protocol satisfies the condition described in relation 1. Detailed information regarding different BC layers can be found in [56].

Note that a generated/received AnoLE message might be sent to all neighbors, which implies that all probable leaders shall eventually know the winner leader if T was sufficient. However, a subset of the network might not have enough time to vote for a leader and receive its MST. This seems OK as nodes use their ONS if available, and broadcast otherwise. With several runs of the AnoLE protocol, and dynamic modification technique of T, T would become more precisely adequate/sufficient. A simple modification technique of T can be defined according to application requirements. For example, nodes may assume that not receiving the MST from the leader they voted for indicates insufficient T. Thus, those nodes may double T for next rounds. On the other hand, receiving the MST sooner than the end of T indicates that T is bigger than needed. Thus, nodes may compute the average of T and the time elapsed from voting till receiving the MST.

The AnoLE protocol utilizes the Epoch time which implies that the location of miners, and distinct transmission delays would not impose a synchronization problem. All nodes thus use the same reference time and all nodes will track T accurately. Hence, all nodes will initiate/terminate the protocol according to unified timestamps.

4.2. Phase-2: Computing and Broadcasting MST

Assuming that T was sufficient for all NPs to vote and for all PLs to receive those votes, we anticipate that by the end of Phase-1, the Leader (L) is recognized by all PLs and by the majority of network miners. Each PL returns to the state 'NP' except for L. Consequently, Phase-2 is triggered and is performed by L as follows:

- Step-1 (Construct NT): L uses its locally saved LVs to construct the anonymized global network topology (NT), represented by an adjacency matrix NT. Algorithm 7 details how L computes NT.
- Step-2 (Compute MST): L utilizes Prim's approach [34] to find MST_{NT}. Note that any other (perhaps better) approach can be utilized here, e.g. [57, 58].
- Step-3 (Broadcast MST): Lastly, the Leader derives its own ONS from the MST it built, as described in Step-2 of Phase-3, and uses it to send the MST to its neighbors in its ONS. The MST is encapsulated in an AnoLE-3 message, which also contains $h(L)$ and the time of MST generation.

4.3. Phase-3: Processing received MST to get ONS

- Step-1 (Verify L): Once a NP receives an AnoLE-3 message, it checks whether this message was gen-

Algorithm 7: Construct Network Topology

```
1  Input h(i);
2  Network_topology = array[[0]];
3  Function ADD_node(node)
4      if NOT node in Network_topology[0] then
5          Network_topology[0].append(node)
6          Network_topology.append([node])
7      end
8  end
9  for LV in self.LVs do
10     if LV[0] == h(i) then
11         ADD_node(LV[1])
12         for neighbor in LV[2][0] do
13             ADD_node(neighbor)
14             node_index_NT =
                Network_topology[0][LV[1]].index
15             neighbor_index_NT =
                Network_topology[0][neighbor].index
16             neighbor_index_LV =
                LV[2][0][neighbor].index
17             weight = LV[2][1][neighbor_index_LV]
18             Network_topology[neighbor][node_index_NT]
                = weight
19             Network_topology[LV[1]][neighbor_index_NT]
                = weight
20         end
21         self.LVs.delete(LV)
22     end
23 end
24 if Network_topology is connected then
25     return Network_topology
26 else
27     return None
28 end
```

Algorithm 8: Derive ONS from MST

```
1  Input AnoLE-3;
2  for key in MST do
3      if key == h(self.id) then
4          return MST[key]
5      end
6  end
```

erated by the leader it previously voted for. The assumption of an adversary node impersonating the real leader is valid. However, such impersonation probability may be solved using asymmetric encryption techniques, where the leader couples a public key with its AnoLE-1 message. Later, the leader can sign the AnoLE-3 message using its private key. This step also implies that the recipient NP is expecting to be present within the proposed MST. Otherwise, this NP will not accept the MST despite it was sent by the leader the NP elected.

- Step-2 (Derive ONS_i): Every miner (including current L and previous PLs) that receives a verified MST (within an AnoLE-3 message) derives its own ONS by running Algorithm 8. The derived ONS is utilized then to optimally select and share data.

- Step-3 (Award L and utilize ONS_i): In case the leader shall be incentivized for its work, the leader may include its wallet id within its AnoLE-1 message. The leader then includes this piece of information within its signature, which adds another layer of verification. Miners which receive the AnoLE-3 message award the leader by adding a predefined amount of digital assets into the leader's wallet. Note that in this case, the AnoLE-3 message also represents a TX that needs not to contain the leader's wallet ID nor its public key because they have already been shared within the AnoLE-1 message.

5. Evaluation

In this section we perform a detailed evaluation of our proposed DONS protocol in terms of security, privacy, time and message complexities, Finality and Fidelity. We compare the AnoLE and the DONS protocols with current methods, and we indicate the strengths, the weaknesses, and open issues of our proposed methods.

Our experiments were carried out on a DELL PC with an Intel i5-8265U CPU (8-Cores, 3.8 GHz) with 12 GB DDR4-SDRAM, 500 GB of SSD and Windows-10 OS.

5.1. Security analysis

Referring to Cachin et al. [59], the following security properties must be guaranteed by a successful distributed protocol:

1. **Strong validity**: If all honest nodes propose the same value v, then no honest node decides a value different from v. This property is indeed guaranteed by the AnoLE and DONS protocols. That is, if all NPs voted for a probable leader k, all probable leaders will announce k as leader. Later, all nodes who voted for k will accept it's proposed MST. The processes that guarantee this property is detailed in Algorithm 1 (lines: 12–14) and Algorithm 6. Further, NPs only vote for a PL who they heard from, which is guaranteed by Algorithm 5.
2. **Agreement**: No two honest nodes decide differently. This property is implicitly guaranteed by the AnoLE and DONS protocols. That is, if T was not sufficient, different PLs may announce different leaders, and different NPs may vote for different PLs. However, only one leader can obtain a majority of votes, leading later to the ineffectiveness of other PLs announcements. The MST computed by PL whom was voted for by the majority of NPs, will be the only MST adopted by this majority. If incentivization was included, the PL who was voted for by the majority will be incentivized by

the majority as well. Accordingly, the decisions of the minority of NPs who voted for, adopted the MST of, and incentivized other PLs, will not be adopted by the network as the majority rule applies in BC systems. Nevertheless, this property can be surely guaranteed if T was sufficient, resulting in all votes arriving to all PLs and, thus, all honest PLs announcing the same leader. The processes that guarantee this property is detailed in Algorithm 1 (lines:12–14) and Algorithm 6.

3. **Termination**: Every honest node eventually decides some value. This property is guaranteed in the AnoLE protocol as each PL changes its status to either L or NP once T has passed. Consequently, any following AnoLE messages received by the node after changing its status shall be ignored. The leader also changes its status to NP once it calculated the MST and shared it with its neighbors. The termination point of the protocol is detailed in Algorithm 1 (lines: 16 and 22). Additionally, all NPs decide which LP they want to vote for, as declared in Algorithm 2 (line: 21). All PLs decide what their next status is, who is the winning PL, and who is their current leader, as declared in Algorithm 6.
4. **Integrity**: No honest node decides twice. This property is guaranteed in the AnoLE protocol for both PLs and NPs, while leaders do not make any decisions. The integrity of decisions made by PLs and NPs are guaranteed using the processes detailed in Algorithm 6 (lines: 11 to 16) and Algorithm 2 (line: 19). Later, every NP shall decide whether to accept or reject a received MST for the current round, depending on the generator. If the the leader who generated the MST was voted for by this NP, the MST is adopted and the leader may then be awarded. This process in declared in Algorithm 1 (lines: 12–13).

A successful distributed ledger shall provide three properties, namely the **Consistency**, the **Availability**, and the **Partition Tolerance**[60]. Mapping these properties onto our DONS security discussion, we state that these issues are not of a concern with the DONS protocol. That is, non consistent MST distribution leads to some NPs performing NS according to randomized selection rather than ONS, which is declared in Algorithm 1 (lines: 2–11). This may negatively affect the overall finality time, yet it has nothing to do with the consistency of the distributed ledger. Furthermore, in the case where leaders are to be incentivized, the original majority rule applies, which implies that if the original BC was consistent before adopting DONS, it will remain consistent after. Similar argument can be stated regarding the Availability issues.

Regarding the Partition tolerance issues, the BC network could be partitioned into two networks if a failing node was a bridge node, which should be solved by the original network architecture, e.g., by requiring a minimum number of neighbor connections upon joining the network. However, it is not harmful to adopt different MSTs by different parts of the network, as this would lead to higher finality time but not a disconnected network (i.e. compared to finality time with unified MST. However, even in such case, DONS shall perform better than RNS and RTT-NS). Note that a disconnected network topology constructed by the leader would not trigger the leader to compute and share the MST, as declared in Algorithm 7 (lines: 24–28). Eventually, NPs who do not receive the MST by the end of T, shall terminate the round and continue using their original NS method (e.g. RNS or RTT-NS). If a minority of NPs receives an MST from another leader, they may use it and incentivize their leader, yet the incentive would not be confirmed by the majority of miners, leading to correct and consistent ledger even with partitioned network.

From another point of view, it could be argued that the utilization of our proposed protocols and the resulting provision of network topology may encourage an **eclipse attack** [61] leading to **DoS** [62] or **Double Spending** [63] attacks. However, Wüst and Gervais [64] described several countermeasures that can be adopted to prevent eclipsing. To address this issue in DONS, we emphasize that each peer in the network maintains its own set of connections, out of which a subset is used to communicate. This is, a peer is not practically isolated from the network and can simply adopt a checker mechanism to secure itself against a logical isolation. Note also that the subset derived from the MST (i.e. ONS) could include more than one, randomly selected neighbor according to the structure of the MST.

A checker mechanism aims at regularly validating the peer's local BC version against neighbors' versions. This way, peers can be sure that they have not been eclipsed by an adversary leader or neighbor. ONS can be simply withdrawn by a peer that has been eclipsed, and it can get back to using its original NS method until a new AnoLE round is triggered. Furthermore, a reputation mechanism, similar to the one adopted by the Proof-of-Stake protocol [65], can be developed.

Next, we discuss the security issues provided by DONS and AnoLE protocols in probable situations that may appear in real-life scenarios:

1. **Problem**: Leader provides an MST that provides the ONS of only a minority (or none) of the network.
   Solution: The puzzle that the leader needs to solve is to find the networks' MST that includes as many network nodes as possible. Accordingly, the proposed MST would be accepted by the majority upon verification. As the puzzle solution is hard to find, the solution is easy to be verified on the NP side by checking if it was included in the proposed MST. Every NP checks whether the MST is proposed by the leader the NP voted for, and whether it is included in the MST. Thus, the MST can be rejected even if it was generated by the leader the NP voted for. Consequently, the more nodes included in the MST, the higher the probability for the MST to be accepted by the majority. This is declared in Algorithm 8 (line: 3).
2. **Problem**: No Leader was announced, which means

each PL announced other PL due to inconsistency in voting distribution.
Solution: Network nodes would keep running using their previous ONS or, in case they had no ONS, a randomized/RTT -based NS. This is declared in Algorithm 1 (lines: 2–11).

3. **Problem**: Multiple Leaders were announced.
Solution: Only one MST will be adopted by the majority of network nodes, as each node only adopts the MST proposed by the Leader it already voted for. If incentives to be granted upon proposing a new MST, the Leader with actual majority number of voters will be incentivized by the majority. Thus, only one Leader will be eventually incentivized by the network. This is declared in Algorithm 1 (lines: 12–14) and Algorithm 2 (line: 21). As PLs know this, no PL node shall dishonestly claim to be a leader node as the work spent to find the MST would not be recognized by the majority. This is declared in Algorithm 1.
4. **Problem**: Round-time T is not sufficient to deliver all votes to leaders.
Solution: Round-time T shall be dynamically modified as would be discussed in Section 6. Furthermore, this would lead to MST proposal that is not inclusive. Accordingly, the leader would not be incentivized. when a node receives an MST that does not include its ONS, it shall automatically increase its default Round-Time T as it would assume that its LV did not have sufficient time to arrive to the leader. Nevertheless, this should not imply a problem as discussed regarding the partition tolerance above.
5. **Problem**: A dishonest PL claimed there was a failing/joining node in order to get incentivized, but the truth is there was no failing/joining node.
Solution: NPs require a minimum number of RCs, as declared in Algorithm 3. These could only be generated randomly and it is unlikely that all neighbors are adversaries and that they collaborate with each other. In any public-permissionless BC, a node is connected, upon joining, to a group of randomly selected neighbors. Accordingly, it is nearly impossible to get all neighbors to agree on the failure of a node that did not fail. However, a dishonest PL may generate several fake AnoLE-1 messages claiming that a node has just joined the network. In this case, NPs assume that the RCs they receive are from nodes that are already existent in the network and they are not just joining. Accordingly, a NP checks the validity of each received RC, by looking for the RC's generator in its previous MST (i.e. the NP's MST). Consequently, RCs are only accepted by nodes who are already proven as nodes of the network. This is declared in Algorithm 3 (lines: 2–10).
6. **Problem**: A dishonest NP provided a faulty LV.
Solution: if hashes of neighbors of this NP are faulty, this NP will not be connected to the network in the proposed MST and will not be able to obtain its ONS. As described in Algorithm 7: lines 12–20. Nevertheless, neighbors of the faulty NP will provide correct LVs as it is unlikely that neighbors collaborate. Accordingly, the ONS the faulty NP would eventually get would be correct. As the fake neighbors claimed by the faulty NP have only been claimed to be connected to by this NP, then these fake neighbors would be leafs in the produced MST and would not affect ONSs of other honest NPs. If hashes were correct but the weights are non-correct, non of the neighbors of this NP will provide similar faulty information, unless the neighbors are collaborating in this. Such collaboration is not possible as described above. The purpose of Algorithm 7 is to solve such problem on the leader side, and for this reason it has the highest time complexity, as will be detailed later.
7. **Problem**: PL cheated and did not wait for randomly selected time after AnoLE trigger.
Solution: This might lead to one or more of the previously discussed problems (specifically 1–4). Accordingly, a cheating PL would not benefit from such behaviour as discussed earlier. However, to guarantee that PLs accurately wait for the selected random waiting time, Trusted Execution Environments (TEE) [66] can be used. Note that this is not mandatory for security reasons but for efficiency guarantees reasons.

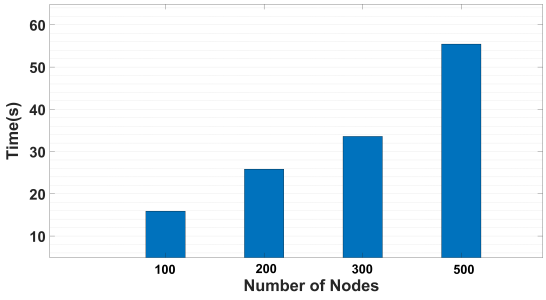

Figure 3: Required time (seconds) for running the AnoLE protocol until delivering a connected MST to nodes of a Barabási-Albert random network with different sizes

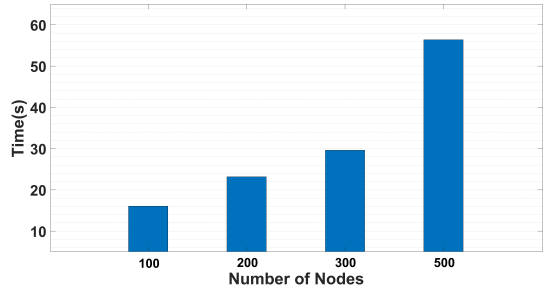

Figure 4: Required time (seconds) for running the AnoLE protocol until delivering a connected MST to nodes of an Erdős-Rényi random network with different sizes

Table 1
Time complexity of each algorithm utilized during the first two phases of the DONS protocol (the AnoLE protocol)

algorithms	Time complexity	
	worst-case(dense graph)	best-case (sparse graph)
algorithm-1	$O(\|V\|^2)$	$\Omega(1)$
algorithm-2	$O(\|V\|^2)$	$\Omega(1)$
algorithm-3	$O(\|V\|^2)$	$\Omega(1)$
algorithm-4	$O(\|V\|)$	$\Omega(1)$
algorithm-5	$O(1)$	$\Omega(1)$
algorithm-6	$O(\|V\|)$	$\Omega(1)$
algorithm-7	$O(\|V\|^3)$	$\Omega(V)$
DONS time complexity	$O(\|V\|^3)$	$\Omega(V)$
DMST	$O(D+\sqrt{\|V\|})$	-

Table 2
Message complexity of each step during the first two phases of the DONS protocol (the AnoLE protocol)

phases	steps	message complexity	
		worst-case(dense graph)	best-case (sparse graph)
phase-1	step-1	$O(\|V\|^3)$	$\Omega(\|V\|)$
	step-2	-	-
	step-3	$O(\|V\|^3)$	$\Omega(\|V\|)$
	step-4	-	-
phase-2	step-1	-	-
	step-2	-	-
	step-3	$O(\|V\|)$	$\Omega(\|V\|)$
DONS message complexity		$O(\|V\|^3)$	$\Omega(\|V\|)$
DMST		$O(\|V\|^2)$	$\Omega(\|V\|)$

5.2. Privacy analysis

Next, we discuss the identity privacy preservation [67] provided by our proposed AnoLE and DONS protocols. As detailed in Sections 2.1 and 2.3, our proposed methods must guarantee the privacy condition (1) proposed in Section 3. Following the description of the proposed protocols in the previous sections, one can notice that data shared between network nodes are exchanged in the form of AnoLE messages. For any given node a, it can receive the three types of AnoLE messages generated by all, or a subset of, network nodes.

The AnoLE-1 message includes the hash of the node i id that left/joined the network, the hash of its neighbor j id, and the timestamp of the message. According to the definition of a hash function provided in Section 3, node a cannot obtain any private information about i or j from AnoLE-1 messages.

The AnoLE-2 message is similar to the AnoLE-1 message, with the addition of the hash of a's id, the hash of the elected leader id, and the Local View of a (LV_a). LV_a consists of the hashes of ids belonging to the neighbors of a, in addition to the RTT a has measured between itself and its neighbors. Thus, any other node b can see that a node with id hash $h(a)$ is connected to a number of neighbors with id

Table 3
Results of the AnoLE protocol simulation experiments on two random network models with different sizes

	Network model							
	BA				ER			
number of nodes	100	200	300	500	100	200	300	500
avg.no.neighbors	2	2	2	5	-	-	-	-
connection probability	-	-	-	-	0.05	0.02	0.015	0.01
default round-time	30	40	60	60	30	40	50	60
DRC	2	2	1	2	2	2	2	2
time(s)	**15.9**	**25.84**	**33.6**	**55.44**	**16.05**	**23.18**	**29.6**	**56.38**
no.of exchanged messages	**676,071**	**3,237,133**	**5,871,463**	**16,380,174**	**788,139**	**3,321,310**	**7,054,606**	**13,617,211**

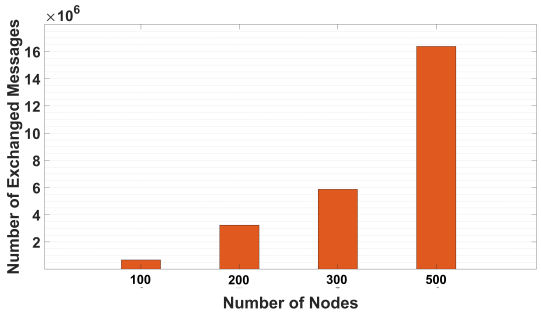

Figure 5: Total number of exchanged messages for running the AnoLE protocol until delivering a connected MST to nodes of a Barabási-Albert random network with different sizes

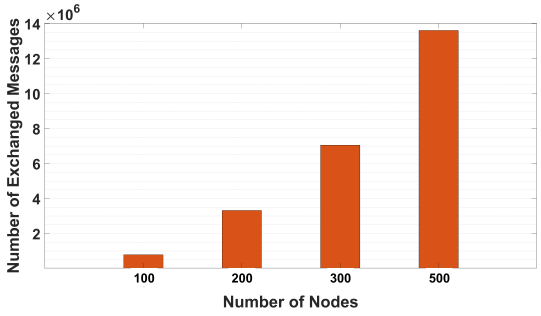

Figure 6: Total number of exchanged messages for running the AnoLE protocol until delivering a connected MST to nodes of a Erdős-Rényi random network with different sizes

hashes $h(1), ..h(n)$ with links of some given weights. However, b cannot determine the true identities of a nor its neighbors, which makes the knowledge of weights on the links useless. In the case where a, or any of its neighbors, is a neighbor to b, b can only determine the true identity of its neighbors.

The last type of exchanged messages is the AnoLE-3 message, which includes the hash of the leader id $h(L)$ and the MST. Note that the MST is a collection of reduced LVs and, thus, what applies to the LV knowledge deduction applies to the MST. Note also, that network nodes accept the anonymized MST and deduce their ONSs by comparison and not by decryption. That is, each node constructs a list of hashes of its neighbors and compares these hashes with hashes in the MST. Additionally, nodes compare the hash of the leader they voted for, with the hash of the AnoLE-3 message generator. If the two were compatible, the MST within the AnoLE-3 message is accepted.

As can be noted from this description, by the end of the protocol run, network nodes can only read the true identities of their neighbors. Additionally, even the leader cannot read any true identity in the MST it builds unless it was for itself or for one of its neighbors. Network nodes further vote for leaders, and accept leaders' MSTs without any knowledge of true identities of leaders.

5.3. Time and message complexity for generating the MST

Next, we evaluate the time and message complexity of the DONS protocol, from the moment when the AnoLE protocol is triggered, until all network nodes are aware of the network's MST (i.e. until the end of Phase-2). Table 1 presents the time complexity of each algorithm utilized in Phase-1 and Phase-2. Table 2 presents the message complexity of each step of the two phases. We compare the final complexities of the first two phases of DONS with the complexities of the method proposed in [18] (notated as DMST). This is because the first two phases of the DONS protocol, which consists of the AnoLE algorithms, aim at providing each node in the network with knowledge about the MST. This objective is similar to the objective of the method proposed in [18]. Following this notation, both the AnoLE protocol and the DMST protocol can be effectively deployed into the DONS protocol. The distinction between the outperformance of the AnoLE protocol compared with the DMST protocol in terms of privacy as discussed in the previous subsection. Higher AnoLE complexities, however, are the cost of a privacy preserving protocol to obtain MST in a our semi-distributed model.

We have implemented the AnoLE protocol using Python 3.8 with utilization of popular packages such as multiprocessing, threading, networkx, hashlib, among others. Our implemented code is publicly available at GitHub[1]. To validate our implementation, we performed several experiments utilizing two random network models, namely Erdős-Rényi (ER) model and Barabási-Albert (BA) model. We oscillated the number of nodes to capture the protocol behaviour within different network sizes. The configuration we used for running our experiments, along with the simulation results, are presented in Table 3 and depicted in Figures 3, 4, 5 and 6.

5.4. Comparison with current methods

In this subsection, we experimentally evaluate the performance of DONS-based BC networks, in terms of block finality time [5] and Fidelity [68]. We compare our results to two NS approaches, discussed previously in Section 2.1, namely randomized NS (RNS) and RTT-based NS (RTT-NS). We utilize two random network models to perform our experiments, namely Erdős-Rényi (ER) model [14] and Barabási-Albert (BA) model [15]. For both models, we oscillate the configuration of network size and average number of neighbors per miner, to demonstrate the consistency of our previous analysis with real-life scenarios.

Specifically, we developed a (Python v:3.8) network simulator, where a random BC network is built and a randomly selected miner represents the source node of a block of data. The generated block is then shared by the source node with a group of neighbors, each of the neighbors shares the block similarly with a group of its neighbors, etc. The simulation terminates once the block reaches all nodes of the network, mimicking the push-based gossiping approach generally adopted by all BC applications. The compared three NS

[1] https://github.com/HamzaBaniata/AnoLE_Protocol

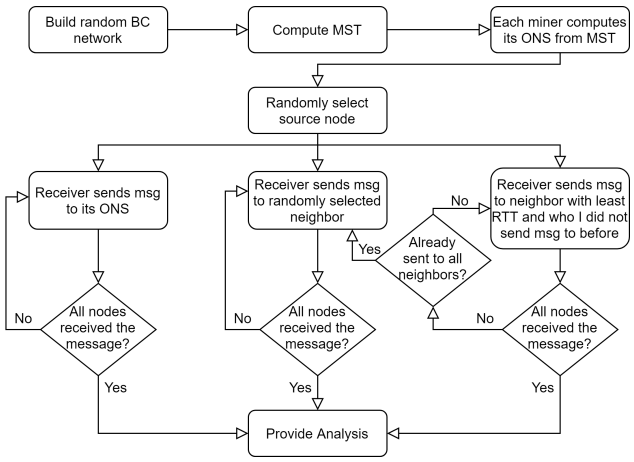

Figure 7: Simulation workflow for push-based gossiping in BCs utilizing DONS, RNS and RTT-NS protocols

methods are utilized consequently using the same network for the same block being generated by the same source node. At the termination of each simulated scenario, total finality time and number of redundant messages are calculated. As miners in public-permissionless BC networks are randomly connected, analyzing the results obtained from running our developed simulator indicates the best NS approach in terms of fidelity and finality. Consequently, we could experimentally prove that the adoption of such NS approach leads to enhanced DL consistency. Our implementation workflow is demonstrated in Figure 7.

Finality is the assurance or guarantee that data cannot be altered, reversed, or canceled after they are completed [69]. To achieve optimal finality in a given BC, shared data needs to be spread as soon as possible through the BC network, so that miners can adopt this data before a new piece of data is generated. The latency level of a BC shall, then, ultimately affect its finality rate. Fidelity, on the other hand, is the degree to which a technique can provide consistency guarantees [68]. To evaluate DONS in terms of fidelity, we count the number of cycles a generated data walks, in a BC network, when utilizing DONS, RNS and RTT-NS. That is, more cycles indicate the overhead on network connection links, overhead in computation at the node level and, accordingly, higher overall finality time. To count the number of cycles a shared data walks, nodes are instructed to count the number of replicated messages they receive. Simultaneously with running the simulation scenarios, a finality-checker process is implemented that regularly checks whether all nodes have yet received the shared data. Once returned a *True* value, all nodes are shut down and the main *Analysis* function is triggered.

We made our simulator code publicly available at a GitHub repository[2]. We run several simulation scenarios, as described in Table 4. The results of our experiments are presented in Table 4 and Figures 8, 9, 10 and 11.

[2]https://github.com/HamzaBaniata/DONS_simulator

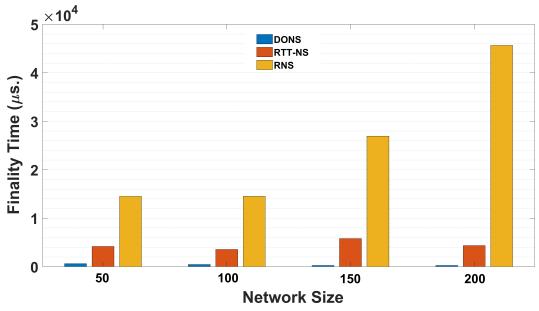

Figure 8: Total Finality time of a randomly generated block by a randomly selected miner, in a Erdős-Rényi (ER) network model (with connection probability = 0.1)

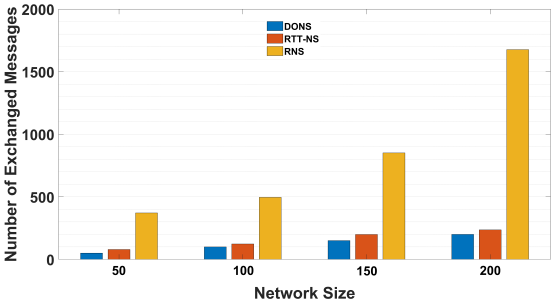

Figure 9: Total number of exchanged messages until a randomly generated block, by a randomly selected miner, is delivered to all network nodes, in a Erdős-Rényi (ER) network model (with connection probability = 0.1)

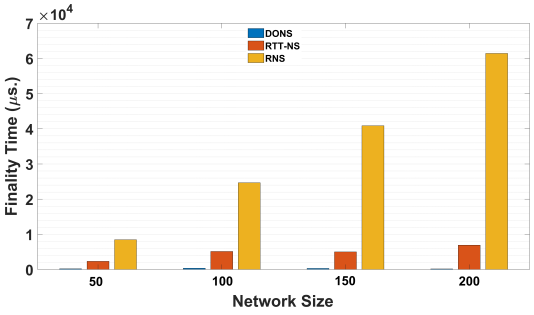

Figure 10: Total Finality time of a randomly generated block by a randomly selected miner, in a Barabási-Albert (BA) network model (with avg. no. of neighbors per miner = 5, 5, 7, 10, respectively)

6. Discussion

In the previous sections, we presented our proposed algorithms and techniques in detail for both DONS and AnoLE protocols, and shared their open-source implementations. We further discussed how our proposed protocols address different misbehaviour situations of system entities in Section 5.1, in order to validate the security of our proposal. In Section 5.3 we compared the AnoLE protocol with a recently proposed method for solving the distributed MSTP, in terms of time and message complexities. Furthermore, we compared the performance of different network models in terms of Fi-

Table 4
Performance of the DONS protocol against RTT-NS and RNS protocols, on two random network models with different sizes

model	number of nodes	Network parameter	Finality time (μs)			Number of exchanged messages		
		avg.no.neighbors/node	DONS	RTT-NS	RNS	DONS	RTT-NS	RNS
BA	50	5	204	2,343	8,463	50	74	253
	100	5	367	5,139	24,667	100	142	751
	150	7	310	5,008	40,870	150	182	1,514
	200	10	176	6,927	61,429	200	251	2,241
		connection probability						
ER	50	0.1	629	4,176	14,510	50	79	371
	100	0.1	450	3,558	14,533	100	123	496
	150	0.1	257	5,795	26,932	150	199	851
	200	0.1	260	4,363	45,656	200	236	1,676

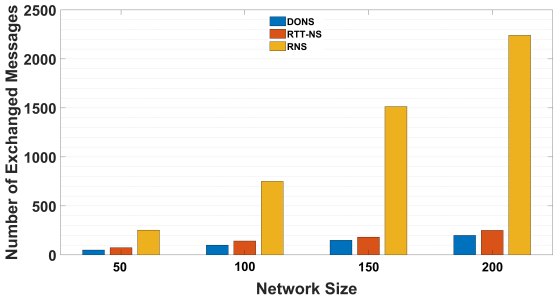

Figure 11: Total number of exchanged messages until a randomly generated block, by a randomly selected miner, is delivered to all network nodes, in a Barabási-Albert (BA) network model (with avg. no. of neighbors per miner = 5, 5, 7, 10, respectively)

nality and Fidelity in Section 5.4, by validating our proposed protocols against earlier methods. Based on the results we can state that our proposed protocols significantly enhance the levels of Finality and Fidelity in the studied networks, due to the provision of globally optimal NS techniques.

Although the time and message complexities for the first two phases of the DONS protocol are higher than those of the recently proposed DMST method, we argue that this shall not be problematic in real-life scenarios. That is, the first two phases of the DONS protocol (represented by the AnoLE protocol) are only triggered, when a node joins or leaves the network. With average miner active time extending to weeks, and even months for many cases, average complexities of the DONS protocol, through long periods of time, shall be decreased. That is, the significant enhancement in terms of Finality and Fidelity once the network nodes utilize the MST, shall positively compensate for the rarely performed high complexities of the AnoLE protocol. Note that in public-permissionless BCs, a new block is generated every few seconds (e.g. Ethereum) or minutes (e.g. Bitcoin), while a node is probably staying online for weeks or months [70]. Furthermore, many miners cooperate in mining pools and warehouses that never leave the network [71].

The adoption of the proposed protocols, then, shall consider the average active time of nodes in the network. That is, relatively rare node failure (e.g. an average active time of nodes equals to a month) implies that the AnoLE protocol is rarely triggered. Accordingly, triggering this protocol may indeed cost much exchanged messages, yet the MST proposed afterwards would definitely enhance data propagation through the network for a long period of time, until a new trigger appears. As a result, adopting our proposed protocols enhances the overall propagation time although it occasionally costs much to find the MST of the new network topology.

Taking as an example, a BA network with size 200 nodes, where the average active time of nodes equals to one week (i.e. 168 hours), and a new block is generated every minute (i.e. 1440 block per 24 hours), we can clarify our last argument. It can be seen in Table 3 that triggering the AnoLE protocol in such network would cost nearly 3.2×10^6 exchanged messages. Once the MST is available to network nodes, they will be able to share their data with a total number of exchanged messages equals to 200 per block of data (Table 4). As a result, the network would exchange a total of $200 \times 1440 \times 7 \approx 2 \times 10^6$ messages per week. Adding this to the 3.2×10^6 exchanged messages to obtain the MST gives a total of almost 5.2×10^6 messages per week. If this network uses the RNS method to share data, the total number of exchanged messages per week would be $1440 \times 2,241 \times 7 \approx 22.6 \times 10^6$ messages per week. Apparently, even with the high rates of exchanged messages by the AnoLE protocol, utilizing it would still be more efficient compared to the currently used methods. The following additional arguments can further be highlighted:

- The AnoLE protocol is one component within the DONS protocol and can be optimized as well as replaced with a better protocol whenever available, leading to even better results.
- As indicated previously, the DMST method can be adopted in BC networks where sharing real identities of nodes is not considered an issue. Such deployment would produce an enhanced DONS protocol in terms of time and message complexities, in addition to being

optimized in terms of finality and fidelity.

- The experimental results presented in Section 5.3 represent the complexities of the AnoLE protocol utilized for the first time. This means that nodes broadcast the messages they receive from their neighbors. However, with multiple leader election rounds run consequently, resulting in larger number of nodes finding their ONSs, total round-time and number of exchanged messages will be significantly decreased as discussed in Section 5.4. Simply put, the results presented in Section 5.3 are the upper bound complexities of the AnoLE protocol.

The potential behind our proposed protocols is apparent for possible extension of current Proof-of-Work (PoW) algorithms into Proof-of-Useful-Work (PoUW) [72]. That is, redefining the puzzle to require finding the MST of the network, instead of solving a mathematical puzzle with no beneficial utility. Such redefinition would maintain the puzzle to be random, fair, verifiable, with an unpredictable solution. Meanwhile, it would definitely enhance the overall throughput and energy consumption of the system. However, more investigations must be carried out for such utilization.

The initial configuration of Round-Time T is highly critical. Very high T value may result in longer times to obtain the needed votes, leading to higher overall time. On the other hand, very low T value may result in limited arrival of LVs leading to exclusive MST, or worse non-connected NT. In such cases, our implementation directs the leader to just not construct the MST. Accordingly, NPs keep using their original NS method (i.e. either a previous ONS or randomized NS). A method to modify the default round time, at the node level, upon the receipt of an MST might be practical. For example, NPs who voted for a leader and have not received the expected MST after T passed, may assume that their current T was not sufficient to perform all steps of the protocol. Accordingly, they double their T value and use the updated value in the next time the AnoLE protocol is triggered. Similarly, NPs that received the expected MST before T passes, may compute the average of: T and time elapsed from the trigger appeared until the MST was received. This way, T is constantly updated according to the size of the network, without NPs being actually aware of the network size. The most recent value of T is used afterwards when the NP is triggered to be a PL (i.e. by a joining or leaving event of one of its neighbors).

The DRC value is only effective while the node has no previous ONS. Once the node obtained its ONS, it can deduce the exact RC value from its previously received MST. We also noted, during the experiments we performed, that higher average number of neighbors initial configuration results in increased latency of AnoLE messages. This is in fact a trivial observation, as broadcasting in random networks leads to more cycles a shared data walks through the network until it is delivered to all nodes. After several rounds of AnoLE and DONS, the number of walked cycles is reduced until it reaches 0 cycles, leading to 0 redundant exchanged messages and 0% fidelity.

We have not provided the detailed algorithm to find the MST as many such algorithms had been proposed in the literature. The algorithm we deployed, i.e. Prim's, has a complexity of $O(V^2)$. This is consistent with the AnoLE protocol complexities as presented in Table 1. Using a more efficient algorithm to find MST, then, does not enhance the overall time efficiency of the AnoLE protocol. However, it shall decrease the energy consumption on the leader side and, consequently, produce the MST faster which affects T.

Although not mentioned in Algorithm 1 for short, we partially adopted a message passing approach inspired by the one proposed by He et al. [33]. A message is not passed to system entities if it has been historically passed to them according to a local log. We use this approach within DONS if the set of selected neighbors was built according to the ONS of the sender. We found this approach not practical when the set of selected neighbors is built according to RNS or RTT-NS methods. For identity privacy reasons, miners in DONS only save messages passed to/from their neighbors, rather than from any other miner in the network as suggested in [33].

7. Conclusion and Future Work

In this paper, we addressed the Neighbor Selection Problem (NSP) for public-permissionless BCs by proposing a Dynamic Optimized Neighbor Selection protocol called DONS. As a first step of the DONS protocol, a leader needs to be elected in order to perform additional computations. To this end, we proposed an Anonymized Leader Election protocol called AnoLE, that aims at electing a leader in a distributed fashion, without any previous knowledge of the network size or nodes private identities. By the end of the AnoLE protocol, a leader is announced to perform the following computations. Meanwhile, neither the elected leader nor the nodes know the identities of each other (except for their original knowledge about their neighbors). The elected leader constructs an anonymized network topology, from which it computes the Minimum Spanning Tree (MST) of the network. An Optimized Neighbor Selection (ONS) is then derived from the MST by the leader and the rest of network nodes, in a private manner. Each node utilizes its derived ONS to communicate with the least number of neighbors, but with optimized communications paths. We analyzed the security and privacy of our proposed protocols, and we provided the time and message complexities of their algorithms. Additionally, we provided publicly available implementations of both protocols, which we used to experimentally validate our approach. Our experiments showed significant enhancement of message propagation for different network models and sizes, in terms of finality and fidelity, compared to similar networks utilizing state-of-the-art methods.

In the future, we plan to investigate the multi-leader scenario and its implications on the security and the efficiency of the DONS protocol. As our current proposal of the AnoLE

protocol does not utilize a compatible privacy-aware leader *incentivization* mechanism, we plan to investigate and deploy a suitable mechanism, and evaluate the trade-offs that need to be tuned. We will focus on some interesting previous works solving similar challenges, e.g. [73, 74, 75, 76, 77]. We also plan to investigate and implement a suitable reputation mechanism compliant with the conditions discussed in Section 5.1. The deployed mechanism shall, of course, adhere to the security measures expected from DONS and AnoLE protocols, e.g. as described in [78]. Finally, we will research the possibility of upgrading the purpose of the DONS and AnoLE protocols into a comprehensive consensus protocol for public-permissionless BCs, turning a PoW-based BC into a PoUW-based BC.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[2] M Sadek Ferdous, M Jabed Morshed Chowdhury, Mohammad A Hoque, and Alan Colman. Blockchain consensus algorithms: A survey. 2020.

[3] Gabriel R Carrara, Leonardo M Burle, Dianne SV Medeiros, Célio Vinicius N de Albuquerque, and Diogo MF Mattos. Consistency, availability, and partition tolerance in blockchain: a survey on the consensus mechanism over peer-to-peer networking. *Annals of Telecommunications*, pages 1–12, 2020.

[4] Furqan Jameel, Muhammad Nabeel, Muhammad Ali Jamshed, and Riku Jäntti. Minimizing forking in blockchain-based iot networks. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[5] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[6] George Suciu, Carmen Nădrag, Cristiana Istrate, Alexandru Vulpe, Maria-Cristina Ditu, and Oana Subea. Comparative analysis of distributed ledger technologies. In *2018 Global Wireless Summit (GWS)*, pages 370–373. IEEE, 2018.

[7] Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. A theoretical model for fork analysis in the bitcoin network. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 237–244. IEEE, 2019.

[8] Minghong Fang and Jia Liu. Toward low-cost and stable blockchain networks. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[9] Robbert Van Renesse, Kenneth P Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM transactions on computer systems (TOCS)*, 21(2):164–206, 2003.

[10] Jon Foss Mikalsen. Firechain: An efficient blockchain protocol using secure gossip. Master's thesis, UiT Norges arktiske universitet, 2018.

[11] Calvin Newport and Alex Weaver. Random gossip processes in smartphone peer-to-peer networks. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 139–146. IEEE, 2019.

[12] Wei Bi, Huawei Yang, and Maolin Zheng. An accelerated method for message propagation in blockchain networks. *arXiv preprint arXiv:1809.00455*, 2018.

[13] Petrică C Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 283(1):1–15, 2020.

[14] Rényi A. Erdős, P. On random graphs,. *Publicationes Mathematicae*, 6:290–297, 1959.

[15] Barabási A.L Albert, R. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[16] Bharat Bhushan, Preeti Sinha, K Martin Sagayam, and J Andrew. Untangling blockchain technology: A survey on state of the art, security threats, privacy services, applications and future research directions. *Computers & Electrical Engineering*, 90:106897, 2021.

[17] AKM Bahalul Haque and Bharat Bhushan. Blockchain in a nutshell: State-of-the-art applications and future research directions. In *Blockchain and AI Technology in the Industrial Internet of Things*, pages 124–143. IGI Global, 2021.

[18] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time-and message-optimal distributed algorithm for minimum spanning trees. *ACM Transactions on Algorithms*, 16(1), 2019.

[19] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

[20] Jia Kan, Lingyi Zou, Bella Liu, and Xin Huang. Boost blockchain broadcast propagation with tree routing. In *International Conference on Smart Blockchain*, pages 77–85. Springer, 2018.

[21] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4): 18–25, 2001.

[22] Huakun Liu, Xin Wei, Ruliang Xiao, Lifei Chen, Xin Du, and Shi Zhang. Oprcp: approximate nearest neighbor binary search algorithm for hybrid data over wmsn blockchain. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):1–14, 2018.

[23] Maleq Khan, Gopal Pandurangan, and VS Anil Kumar. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(1):124–139, 2008.

[24] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[25] Kaushik Ayinala, Baek-Young Choi, and Sejun Song. Pichu: Accelerating block broadcasting in blockchain networks with pipelining and chunking. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 221–228. IEEE, 2020.

[26] Robbert van Renesse. A blockchain based on gossip?-a position paper. *Cornell University*, 2016.

[27] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[28] Ke Wang and Hyong S Kim. Fastchain: Scaling blockchain system with informed neighbor selection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 376–383. IEEE, 2019.

[29] Yusuke Aoki and Kazuyuki Shudo. Proximity neighbor selection in blockchain networks. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 52–58. IEEE, 2019.

[30] Ming Jin, Xiaojiao Chen, and Sian-Jheng Lin. Reducing the bandwidth of block propagation in bitcoin network with erasure coding. *IEEE Access*, 7:175606–175613, 2019.

[31] Bin Yu, Xiaofeng Li, He Zhao, and Tong Zhou. A scalable blockchain network model with transmission paths and neighbor node subareas. *Computing*, pages 1–25, 2021.

[32] Jiao Li. Data transmission scheme considering node failure for blockchain. *Wireless Personal Communications*, 103(1):179–194, 2018.

[33] Xiaowei He, Yiju Cui, and Yunchao Jiang. An improved gossip algorithm based on semi-distributed blockchain network. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 24–27. IEEE, 2019.

[34] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[35] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[36] Gopal Pandurangan, Peter Robinson, Michele Scquizzato, et al. The distributed minimum spanning tree problem. *Bulletin of EATCS*, 2 (125), 2018.

[37] Otakar Borvka. O jistém problému minimálním. 1926.

[38] Farhad Soleimanian Gharehchopogh and Hassan Arjang. A survey and taxonomy of leader election algorithms in distributed systems.

Indian Journal of Science and Technology, 7(6):815, 2014.

[39] Ittai Abraham, Danny Dolev, and Joseph Y Halpern. Distributed protocols for leader election: A game-theoretic perspective. In *International Symposium on Distributed Computing*, pages 61–75. Springer, 2013.

[40] Mohammed Al Refai. A new leader election algorithm in hypercube networks. In *Symposium Proceedings*, volume 2, 2006.

[41] Mohammed Al Refai. Leader election algorithm in hypercube netwok when the id number is not distinguished. *Information & Communication Systems*, pages 229–237, 2011.

[42] Amit Biswas, Ashish Kumar Maurya, Anil Kumar Tripathi, and Samir Aknine. Frlle: a failure rate and load-based leader election algorithm for a bidirectional ring in distributed systems. *The Journal of Supercomputing*, 77(1):751–779, 2021.

[43] Koji Nakano and Stephan Olariu. A randomized leader election protocol for ad-hoc networks. In *SIROCCO*, pages 253–267, 2000.

[44] Navneet Malpani, Jennifer L Welch, and Nitin Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 96–103, 2000.

[45] Gang Zhang, Jing Chen, Yu Zhang, and Chungui Liu. Research of asynchronous leader election algorithm on hierarchy ad hoc network. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2009.

[46] Alaa N Alslaity and Sanaa A Alwidian. A k-neighbor-based, energy aware leader election algorithm (kelea) for mobile ad hoc networks. *International Journal of Computer Applications*, 975:8887, 2012.

[47] Vaskar Raychoudhury, Jiannong Cao, and Weigang Wu. Top k-leader election in wireless ad hoc networks. In *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, pages 1–6. IEEE, 2008.

[48] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 305–319, 2014.

[49] David Yakira, Avi Asayag, Gad Cohen, Ido Grayevsky, Maya Leshkowitz, Ori Rottenstreich, and Ronen Tamari. Helix: A fair blockchain consensus protocol resistant to ordering manipulation. *IEEE Transactions on Network and Service Management*, 2021.

[50] Teik Guan Tan, Vishal Sharma, and Jianying Zhou. Right-of-stake: Deterministic and fair blockchain leader election with hidden leader. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2020.

[51] Dariusz Dereniowski and Andrzej Pelc. Leader election for anonymous asynchronous agents in arbitrary networks. *Distributed Computing*, 27(1):21–38, 2014.

[52] Daniel A Marcus. *Graph theory*, volume 53. American Mathematical Soc., 2020.

[53] CISSP Thomas Porter, CCDA CCNP, Michael Gough, et al. *How to cheat at VoIP security*. Syngress, 2011.

[54] Leighton Johnson. *Security controls evaluation, testing, and assessment handbook*. Academic Press, 2019.

[55] Anichur Rahman, Md Jahidul Islam, Antonio Montieri, Mostofa Kamal Nasir, Md Mahfuz Reza, Shahab S Band, Antonio Pescape, Mahedi Hasan, Mehdi Sookhak, and Amir Mosavi. Smartblock-sdn: An optimized blockchain-sdn framework for resource management in iot. *IEEE Access*, 9:28361–28376, 2021.

[56] Yong Yuan and Fei-Yue Wang. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(9):1421–1428, 2018.

[57] Bernard ME Moret and Henry D Shapiro. An empirical analysis of algorithms for constructing a minimum spanning tree. In *Workshop on Algorithms and Data Structures*, pages 400–411. Springer, 1991.

[58] Michael L Fredman, Robert Sedgewick, Daniel D Sleator, and Robert E Tarjan. The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1(1-4):111–129, 1986.

[59] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. *Introduction to reliable and secure distributed programming*. Springer Science & Business Media, 2011.

[60] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[61] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.

[62] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, pages 601–619, 2020.

[63] A Begum, AH Tareq, M Sultana, MK Sohel, T Rahman, and AH Sarwar. Blockchain attacks, analysis and a model to solve double spending attack. *International Journal of Machine Learning and Computing*, 10(2):352–357, 2020.

[64] Karl Wüst and Arthur Gervais. Ethereum eclipse attacks. Technical report, ETH Zurich, 2016.

[65] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[66] Yubin Xia, Zhichao Hua, Yang Yu, Jinyu Gu, Haibo Chen, Binyu Zang, and Haibing Guan. Colony: A privileged trusted execution environment with extensibility. *IEEE Transactions on Computers*, 2021.

[67] Dev Arora, Siddharth Gautham, Harshit Gupta, and Bharat Bhushan. Blockchain-based security solutions to preserve data privacy and integrity. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 468–472. IEEE, 2019.

[68] Jiang Lan, Xiaotao Liu, Prashant Shenoy, and Krithi Ramamritham. Consistency maintenance in peer-to-peer file sharing networks. In *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*, pages 90–94. IEEE, 2003.

[69] Emmanuelle Anceaume, Antonella Pozzo, Thibault Rieutord, and Sara Tucci-Piergiovanni. On finality in blockchains. *arXiv preprint arXiv:2012.10172*, 2020.

[70] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 358–367. IEEE, 2016.

[71] Behavioral mining: Blockchain's new rocket fuel (part 1), Sep 2018. URL https://medium.com/the-notice-board/behavioral-mining-blockchains-new-rocket-fuel-part-1-ca65a7d82d22.

[72] Hjalmar K Turesson, Henry Kim, Marek Laskowski, and Alexandra Roatis. Privacy preserving data mining as proof of useful work: Exploring an ai/blockchain design. *Journal of Database Management (JDM)*, 32(1):69–85, 2021.

[73] Huiying Xu, Xiaoyu Qiu, Weikun Zhang, Kang Liu, Shuo Liu, and Wuhui Chen. Privacy-preserving incentive mechanism for multi-leader multi-follower iot-edge computing market: A reinforcement learning approach. *Journal of Systems Architecture*, 114:101932, 2021.

[74] Xin Wang, Jianping He, Peng Cheng, and Jiming Chen. Privacy preserving collaborative computing: Heterogeneous privacy guarantee and efficient incentive mechanism. *IEEE Transactions on Signal Processing*, 67(1):221–233, 2018.

[75] G Suriya Praba Devi and JC Miraclin Joyce Pamila. Accident alert system application using a privacy-preserving blockchain-based incentive mechanism. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 390–394. IEEE, 2019.

[76] Jiejun Hu, Kun Yang, Kezhi Wang, and Kai Zhang. A blockchain-based reward mechanism for mobile crowdsensing. *IEEE Transactions on Computational Social Systems*, 7(1):178–191, 2020.

[77] Osamah Ibrahim Khalaf and Ghaida Muttashar Abdulsahib. Optimized dynamic storage of data (odsd) in iot based on blockchain for

wireless sensor networks. *Peer-to-Peer Networking and Applications*, pages 1–16, 2021.

[78] Shivam Saxena, Bharat Bhushan, and Mohd Abdul Ahad. Blockchain based solutions to secure iot: background, integration trends and a way forward. *Journal of Network and Computer Applications*, page 103050, 2021.