The Rated List: A Sybil-Resistant Framework for Data Availability Sampling in Distributed Hash Tables

Sam Evans
University of Amsterdam
sam.evans@os3.nl
Supervised by: Onur Ascigil, *Lancaster University*

***Abstract*— This paper investigates how a novel reputation-based, hierarchical security mechanism can improve the performance of the Data Availability Sampling process in Distributed Hash Tables while under large-scale sybil attacks. The security mechanism was implemented in Java as an extension to Datahop's existing kademlia-simulator program. Testing and evaluation of the mechanism under worst-case conditions shows resilience to sybil attacks at up to 40% sybil proportion. However, the tests also reveal challenges in scaling the mechanism effectively with the increasing size of the network. While the mechanism improves performance up to 25% in a network of 300 nodes, it degrades performance in a network of 2000 nodes. This paper evaluates the performance of the mechanism under challenging conditions and offers a theoretical basis for its improvement.**

Index Terms: **Data Availability Sampling, Distributed Hash Tables, Kademlia, Ethereum Proof-of-Stake**

I. Introduction

As Ethereum continues to gain widespread adoption, it faces significant challenges, particularly in scalability. While a payment processor like Visa maintains around 1700 Transactions Per Second (TPS), Ethereum oscillates around 14 [11]. Due to inherent scalability limitations, this rate has remained relatively unchanged over time. As the number of users on the Ethereum platform increases, its ability to process all their transactions diminishes, leading to network congestion and elevated gas fees. This highlights the need for effective scalability solutions - one of which is the implementation of transaction rollups.

Transaction rollups are created off-chain and aggregate multiple transactions into a single batch. Validators check rollups for availability, process them, and add them to the blockchain as a single transaction. This approach significantly reduces the amount of data that the Ethereum blockchain needs to process at any one time; however, ensuring the validity and availability of these rollups is crucial for maintaining the integrity and security of the network.

One method for validating transaction rollups is Data Availability Sampling (DAS). DAS is a peer-to-peer mechanism in which a rollup is distributed across a Distributed Hash Table (DHT). Nodes that do not hold their required samples of the rollup are split into two subgroups. Each node in the first subgroup samples two random rows and two random columns of the rollup. In the second subgroup, each node samples roughly 75 random points in the rollup. While nodes are called validators, the purpose of these checks are to ensure availability alone and nodes make no effort to validate transactions on the blockchain. Instead, available data can be used later to validate and detect fraudulent transactions conducted by malicious actors - this is not a function of Data Availability Sampling.

The underlying assumption of DAS is that the entire rollup is available if several nodes can successfully sample completely random points in the rollup. In Ethereum, the validation period for a rollup is set at 4 seconds; therefore, successful verification of the rollup's availability relies on the sampling process' efficiency and effectiveness.

Despite DAS's potential to enhance scalability, one risk that has not been explored in detail is the impact of sybil attacks on the validation process. Sybil attacks involve malicious actors creating multiple fake identities to manipulate the network. In the context of DAS, sybil nodes could simultaneously deny access to requested rollup samples, preventing block validation and delaying submission. This is called a data withholding attack.

This paper addresses this gap by defining and demonstrating a proof-of-concept for a sybil-resistant DAS mechanism. This project's contribution includes a detailed definition of the proposed mechanism and a practical validation of its effectiveness in mitigating the risks posed by sybil-based data withholding attacks. By enhancing DAS's resilience against such attacks, the mechanism aims to ensure more reliable and secure transaction rollup validation, thereby contributing to the overall scalability and robustness of the Ethereum platform.

A. Research Questions

The following research question has been defined to direct the project's approach:

***RQ1:* To what extent does the proposed mechanism improve DHT robustness to sybil attacks?**

With the following sub-question:

- ***RQ1.1*** Is it feasible for a DHT using the proposed mechanism to recover from a large-scale sybil attack in the same validation period in which the attack occurs?

B. Paper Format

The rest of the paper is structured as follows: Section II outlines the theoretical basis for the project, Section III outlines previous work related to this project, Section IV explains the proposed mechanism and the implementation of it in practice. Section V describes how the results were collected, and sections VI and VII show and explain the results. Conclusions are drawn from the findings in section VIII.

II. Background

A. Blockchain

A blockchain is a peer-to-peer, append-only Distributed Ledger. These data structures permit users who do not trust each other to interact without a trusted third party. Blockchain's name stems from the fact that each entry (block) in the Distributed Ledger is cryptographically bound (chained) to those before it. As a consequence, a modification of any previous block invalidates the cryptographic seal on each subsequent block. This cryptographic seal ensures the integrity of all data in the blockchain.

The process of adding blocks to a blockchain is challenging. Because there is no central authority, once a new block is added, blockchain network members must reach a consensus on the chain's state before it can continue. The most common consensus methods in cryptocurrencies like Ethereum are *proof-of-work* and *proof-of-stake*.

Proof-of-work is a highly resource-intensive consensus algorithm widely used in blockchain. The algorithm necessitates that participants, known as verifiers or miners, engage in repeated and complex cryptographic operations to identify a specific number known as a nonce (number only used once). The nonce must satisfy predefined conditions that ensure the validity and security of the new block being added to the blockchain. These cryptographic operations typically involve hashing functions, where miners iteratively alter the nonce and hash the result until they discover a hash value that meets the network's difficulty target. This process requires substantial computational power and energy consumption as it involves solving problems intentionally designed to be computationally challenging - but easy to verify.

Because of its cost, *proof-of-work* consensus algorithms represent a scalability limitation in blockchain, highlighting the need for effective alternatives. New implementations propose the use of *proof of stake* in which nodes pay a deposit to the blockchain to become "verifiers". In Ethereum's case, nodes must pay 32 ETH to become validators, equivalent to €103,099 at the time of writing. Once an entrance deposit is paid, a node may contribute to the consensus mechanism as an equal. The purpose of the fee is twofold: first, nodes prove that they have a vested interest in the network by proving investment, and second, nodes stake an amount that can be used to ensure honest cooperation in the validation process.

Another method by which the scalability limitations of Ethereum, in particular, have been addressed in recent years is by the introduction of transaction rollups. Transaction rollups are constructed off-chain and contain a number of transactions to be added back into the blockchain as a batch. The purpose of the process is to reduce the computational burden placed upon the blockchain and to allow external platforms to handle the transactions themselves. Rollups have a relatively straightforward construction and, for the purposes of this project, can be considered simply a matrix of transaction entries each called a "cell".

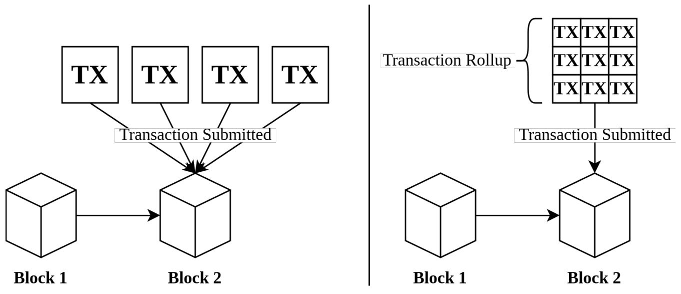

Fig. 1. Adding transactions to a new block, without (Left) or with (Right) a Transaction Rollup

B. Data Availaiblity Sampling

DAS is a method by which nodes can verify the availability of an entire rollup without inspecting each transaction individually. A rollup with 512x512 entries, each entry being 512 bytes, is around 130MB. Without sampling, this must be distributed to a validation network of (in Ethereum's case) around 900,000 nodes and checked for availability before the 4-second validation period passes - this is extremely difficult given current technical limitations.

To improve scalability, the DAS process is completed in a DHT. A DHT is a decentralised and distributed system that provides scalable and efficient data storage and retrieval across an extensive network of nodes. Unlike a usual Hash Table, in DHTs data is distributed across multiple nodes and is routed and looked up using algorithms like Chord [9], Kademlia [6] and Pastry [8]. In this mechanism, nodes may not enter the DAS DHT unless they have been introduced. An introduction involves bootstrapping a new node's known peers so that they may contact nodes in the DHT other than their immediate introducer.

The process of DAS for the purposes of this project has two phases: validator seeding and validator sampling.

1) Validator Seeding: Validator seeding is the process by which portions of a rollup are distributed in a network. A full rollup is given to a resourceful builder node, usually a datacentre node with high up/downlink capability and processing power. Builders take sets of individual cells in the rollup, called parcels, and use a well-known algorithm to determine where the parcel will be placed or assigned in the DHT's ID space. Validators who occupy the parcel's now calculated ID space will be "seeded" the parcel by the builder - which effectively means that the validators will now hold the parcels themselves.

2) Validator Sampling: Validators attempt to acquire two rows and two columns of the transaction rollup. If they must acquire a cell in the transaction rollup that they do not hold, they must find the validator that holds it. In this example, we assume that all nodes know each other. The validator will use the well-known cell-ID space mapping to find one or more seeded validators who have been assigned a parcel which contains the required cell. The validator can then request the cell from the seeded validator - which, assuming it is not behaving maliciously, will return the cell. This process is called a sampling operation and occurs many times throughout the validator sampling process.

C. Sybil Attacks

A sybil attack (sometimes called "sock-puppeting") is a type of attack in distributed systems in which a threat actor creates multiple fake identities to manipulate the network (often to the their benefit). In DAS, sybils can manipulate the network through a range of different attacks; however, this paper will focus on the "data withholding attack."

A data withholding attack is a denial-of-service attack carried out by sybils disrupting the DAS process. It is conducted by an arbitrary number of coordinated sybil nodes mimicking benign operations until a coordinating threat actor orders an attack. When an attack is instructed, all sybil nodes fail to respond to sampling operations. Consequently, nodes that request cells from the sybils will not have those requests fulfilled.

The consequences of a data withholding attack can significantly damage the DAS process. Validators who attempt to sample a malicious node will be forced to make additional sampling operations to other validators who hold the required cell. This process takes time, and if the amount of time taken to recover from the sybil attack exceeds the 4-second validation period, the network cannot come to a consensus on the availability of the transaction rollup in time.

A threat actor may have several reasons to prevent consensus from being reached for any individual rollup. Firstly, a threat actor may want to cause as much damage as possible to the blockchain network - denying service to sampling operations may cause the validation period to pass without consensus and, therefore, deny the rollup's submission to the blockchain. Alternatively, as submitted transactions make money for the submitter, a threat actor may find a particularly profitable transaction within the transaction rollup and attempt to submit it before the rollup is submitted to earn the gas fee from the transaction. They can increase their chances of success by conducting data withholding attacks against the DAS process for that specific rollup, thereby potentially delaying its submission.

While dangerous, prohibitive *proof-of-stake* requirements make wide-ranging sybil attacks unfeasible in the Ethereum blockchain's current form. To hijack the consensus mechanism by pure numbers (assuming no sybils currently exist) would require 900,000 sybils to be submitted (therefore, 28.8M ETH—€92Bn at the time of writing). While this is true, it is valuable to understand how an extensive sybil attack may have consequences on the DAS process and the whole blockchain.

Detecting and removing sybil nodes from a validator's known neighbours is challenging because the beginnings of a data withholding attack closely resemble network congestion. If a protective mechanism were to exclude validators who do not respond to a single sampling operation, it could cause more harm than good. This is because validators are just nodes on a network; a busy node may not be able to respond to all sampling operations in time. An overly aggressive validator pruning algorithm may, in practice, cause a validator's pool of sampling candidates to become so small that some samples cannot be acquired entirely.

III. Related Work

Several works have explored various aspects of DAS and related technologies, focusing on improving security, efficiency and robustness in DHTs and blockchain networks.

Considering data availability, Yu et al. (2020) [12] introduce the Coded Merkle Tree, a novel hash accumulator designed to protect against blockchain data availability attacks. Coded Merkle Trees ensure efficient verification of full availability through downloading an $\theta(1)$ block hash commitment and random sampling. The authors prove that nodes can determine any tampering of the coded Merkle tree with only one honest node connected to the system. The work is a benchmark in the practical evaluation of data availability and represents a possible goal for future data availability verification approaches.

In DHTs, dependable and secure routing algorithms have been introduced to improve resilience against sybil attacks. Routing algorithms such as Maymounkov & Mazières' Kademlia (2002) [6], Stoica et al.'s Chord (2003) [9] and Rowstron & Druschel's Pastry (2001) [8] were designed with reliability and scalability in mind first. These have been expanded to be more secure.

In 2007, Baumgart and Mies propose S/Kademlia [2]. S/Kademlia proposes a secure key-based routing protocol resilient against common attacks like sybil attacks by injecting heavy computation requirements into routing. The paper evaluates the security of the proposed extensions to Kademlia and shows promising improvements under challenging attack conditions. Such work was considered closely when determining how the mechanism should be implemented in this project, though it was eventually discarded as part of a move away from proof-of-work-like approaches requiring high computational requirements.

Similarly, in 2016, Riccardo Pecori [7] introduced S-Kademlia. S-Kademlia proposes a trust and reputation-based algorithm to determine reliable and legitimate nodes. The paper finds varying degrees of performance given differing proportions of sybil nodes - showing promising results, especially under smaller-scale attacks. Such reputational approaches were integrated into the mechanism implemented in this project.

Peer sampling has also been explored by several papers, some of which are separate from blockchain. SecureCyclone, written by Antonov and Voulgaris in 2023 [1], proposes a novel approach to peer sampling designed to eliminate malicious overrepresentation in peer-to-peer networks. SecureCyclon redefines node descriptors into communication certificates and implements a parent-child relationship. The paper shows SecureCyclon's resilience at up to 40% malicious nodes. The parent-child relationship in this approach has also been integrated into the mechanism implemented in this project.

In blockchain, scalability solutions have also been explored in detail. Thibault et al. (2022) [10] evaluate how different

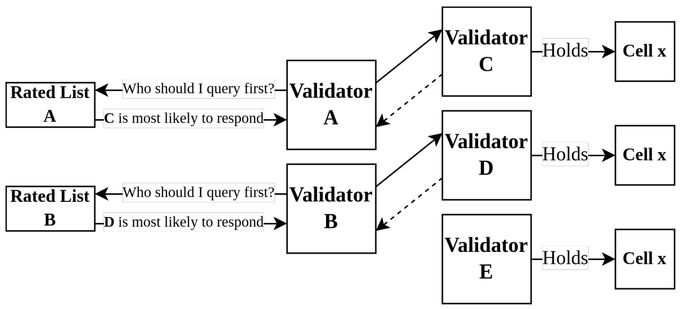

Fig. 2. Each validator with their own Rated List, making their own selection on three validators holding the same sample (X)

types of rollups can be implemented in blockchain and the potential challenges associated with current rollup solutions. The paper underscores the need for ongoing research to address current challenges preventing the widespread adoption of rollups.

Król et al. (2023) [5] evaluate how DAS can be implemented in Ethereum, identifying both network requirements and the challenges associated with specific implementations in Ethereum itself. The paper identifies this project as a potential research direction.

Work relating to sybil attacks, defined by Douceur et al. in 2002 [4], has also been explored in detail. Danezis et al. (2005) [3] published a seminal work on sybil resistance in DHTs. They introduced using a bootstrap graph to track each peer's origin. This is a concept used in the Rated List, an unpublished at the time of writing whitepaper by Dankrad Feist, which was used as the principle concept used in this paper's implementation.

IV. Mechanism: The Rated List

The Rated List, proposed by Dankrad Feist in their whitepaper, is a proposed security mechanism to be used in conjunction with the existing DAS framework. The paper proposes the Rated List and evaluates its scalability in the network. This project implements a modified form of the Rated List as an extension[1] to an existing DAS simulator[2] using the Kademlia DHT implementation.

The Rated List allows an individual validator to maintain a perspective on the reliability of known validators. Nodes perceived as reliable earn higher ratings than those not perceived as reliable. Because each validator has its own rated list, their perspectives on the status of network members can differ - for example, two validators, A and B, can have differing views on the reliability of a third validator, C, known to them both. While A and B have made the same number of requests to C, C has responded to all of A's but only half of B's. As a result, B sees C as a less reliable validator from which to request samples and may attempt to avoid C if a better alternative is possible. This is visualsed in figure 2.

The Rated List is used to decide which validator to sample from. Consider that validator A knows that a required cell is held in validators B, C, and D. A wants to minimise the time taken to acquire the cell so that they can complete their sampling operations as quickly as possible (and, therefore, enable a consensus to be reached as soon as possible). This means that A wants to send a sampling operation to the validator who is most likely to respond first. A asks its Rated List to order nodes B, C and D in order of most (perceived) reliable to least. A then queries B, C and D in the order returned - aiming to minimise the time spent re-sending sampling operations that go unanswered.

[1]github.com/samevans77/kademlia-simulator

[2]github.com/datahop/kademlia-simulator

A. Mechanism Construction and Modification

The Rated List is constructed using mechanisms inherent to the already-existing DAS framework. If a sampling operation is replied to, the replying node will also return a list of its peers. This allows a validator who may only know one or two other validators to build out their "search table": the set of peers the validator has discovered and their location. When replying node A's set of peers [B] is received, the Rated List considers all peers in [B] children of A. The purpose of this parent/child relationship is to keep track of where potentially malicious (and reliable) nodes may come from - this allows the Rated List to change a parent's Rating based on its introduced child's actions.

The Rated List determines the reliability of validators by assigning each one a rating that fluctuates depending on their performance. When a validator successfully replies to a sampling operation, it and all of its ancestor's ratings receive a linear increase. The increased amount can be configured in the mechanism itself, and a validator's rating cannot exceed a specified maximum amount.

When a validator fails to reply to a sampling operation, its and all of its ancestors' ratings decrease exponentially (by base b, in this case). The rating decrease (d) is calculated as a function of the maximum tolerated number of consecutive failures ($maxFail$) and the maximum rating such that:

$$d = \sum_{k=0}^{maxFail-1} F_0 \cdot b^k$$

Where F_0 is the rating decrease caused by the first consecutive failure, calculated by:

$$F_0 \geq maxRep \cdot \frac{b-1}{b^{maxFail}-1}$$

The base of the exponent (b), the number of maximum fails ($maxFail$), and the maximum reputation ($maxRep$) are all configurable - and the rated list then calculates an F_0 to satisfy the above requirements.

The algorithm has been designed to limit the first reduction in reputation but ensure that $maxFail$ consecutive failures cause a node with a maximum rating to fall to zero rating. It is also designed to allow the severity of this first reduction to be configurable through the base. For a lower initial rating reduction, increase the base.

When the Rated List receives a list of candidate nodes to order, it acquires each node's ratings and returns the candidates in order from highest reputation to lowest.

B. Scalability

The mechanism is designed to be scalable by limiting its structure. Parents are limited to a maximum number of children. In the implemented mechanism, the number of children is limited to the maximum number of peers returned from a sampling operation—16. The "root" is allowed 16 children who act as top-level ancestors. Furthermore, to limit computational complexity in calculating parental paths, the maximum number of levels of ancestors is limited to three. That means a top-level ancestor can have no more than three "generations" of children. Furthermore, nodes may not appear in the Rated List more than once and may only have a parent as a direct ancestor in the Rated List structure. In this implementation, the total number of peers that can be held in the Rated List is $16^4 = 65536$. In the mechanism implemented, the maximum tolerated depth and number of children are straightforward to adjust to conduct further experiments.

Consider the example: Node A has children B and C, and C has a child D. If D returns a sample request with B as a peer, it is impossible to record D as B's parent, so the relationship is rejected from the Rated List. Similarly, if A returns peer D, it is impossible to add them as a direct parent - nor is it possible to add D again to the Rated List, so the relationship is rejected.

Calculations of the scalability of a real-life Rated List implementation in Ethereum are conducted by Feist in his whitepaper. The whitepaper shows that a practical implementation would be feasible, costing limited additional bandwidth given the current Ethereum network.

C. Additional Features

In their whitepaper, Feist also discusses potential problems with the Rated List, as well as possible remediation strategies. One problem is the following: validator A must acquire a cell from one of nodes B, C, or D. All nodes have the same rating. C and D have the same parents in this case - while B has different parents. A's Rated List decides that it is more likely for a sybil to have the same set of parents than a legitimate node - as sybils must be introduced to the network by a likely malicious co-conspirator. Therefore, A picks node B, the node with the most diverse set of parents.

This was implemented in the mechanism but did not work as intended, reducing overall performance. The code is written, and a testing mechanism is in place, to combine the node's ratings and diversities as necessary. The parental relationship imposed by the Rated List may be too abstract to include the diversity of the parents currently, and more investigation into the reason for the reduction in performance of diverse parent sampling may be necessary.

V. Method

The mechanism was created and tested as an extension to the kademlia-simulator Java program developed by datahop[3]. Tests were conducted over five simulated minutes of continuous sampling operations with 300 or 2000 nodes in the network to determine current Rated List structure optimisations.

[3]github.com/datahop/kademlia-simulator

Tests were conducted with varying proportions of sybils in the network. The attack is assumed to be worst-case: sybils coordinate perfectly, each denying service simultaneously. Sybils also attempt to saturate legitimate validator search tables (and therefore Rated Lists) with more sybil nodes. This is done by modifying the response to a sampling operation during legitimate operation. A legitimate validator would return all peers that it knows along with the sample, and this list can include a mix of both sybils and legitimate nodes. This is because the validator itself does not know the legitimacy of its peers. A sybil node is assumed to know which of its peers are also sybils and is programmed to return those sybil validators only.

All tests have two parts. In the first part—in this case, the first 2.5 minutes—all sybils behave as normal legitimate nodes (apart from returning only sybils in sampling operations). Once the simulation reaches (in this case) 2.5 minutes, all sybils begin the data withholding attack, failing to respond to all cell requests. These requests will eventually time out; at this point, legitimate validators will attempt to find another validator which holds the cell they require.

The simulator precisely records all operations during the simulation and outputs the process to a log file. To benchmark the performance of the proposed mechanism, the outputted log files are processed, and the network's performance is evaluated. The observed values are:

1) **Average Time to Complete a Sampling Operation:** This metric evaluates the mechanism's ability to improve the network's ability to reach a consensus within the required 4-second validation window.
2) **Average Sampling Operation Completion Rate:** This metric evaluates the proportion of sampling operations which succeed; this allows evaluation of the mechanism's performance in modifying the sampling process.

The recorded evaluation of the network does not begin until after the attack time; this means that the recorded results are only of the attacked network and the network's response to an attack, not of the network's performance under benign conditions. The time before the attack can be considered "network construction time"; in a real-world scenario, this can be any length of time during which sybils behave as expected until an attack is ordered. Again, this makes the attack a worst-case situation for the security mechanism.

Several Python scripts evaluate the network, searching the Java project's output for relevant operations and timings.

VI. Results

Table I shows the set of all results collected with 300 nodes in the network, and table II shows the set of all results collected in a network with 2000 nodes. They show the Average Completion Time (*Avg. Comp. Time*) and Percentage of Completed Operations (*% Comp. Ops*) for each of the tolerated Maximum Number of Failures (*Max Fail*) tested in the experiments as established in section IV-A. This data is visualised for 300 nodes in figures 3 and 4, and for 2000 nodes in figures 5 and 6. The data shows a baseline data point (B), followed by data points for 3, 2, and 1 maximum tolerated consecutive failures.

TABLE I
300 NODE NETWORK

	10% Sybil Nodes				**20% Sybil Nodes**			
Max Fail	B	3	2	1	B	3	2	1
Avg. Comp. Time	1387.0	1152.8	1113.5	1087.1	2679.6	2282.4	2170.2	2205.3
% Comp. Ops	85.3	88.3	88.6	88.8	76.9	80.3	81.2	80.8

	30% Sybil Nodes				**40% Sybil Nodes**			
Max Fail	B	3	2	1	B	3	2	1
Avg. Comp. Time	4276.8	3222.2	3238.8	3280.6	5477.0	5049.1	5205.2	5264.1
% Comp. Ops	65.1	73.5	73.4	73.0	54.8	58.6	57.3	56.8

TABLE II
2000 NODE NETWORK

	10% Sybil Nodes				**20% Sybil Nodes**			
Max Fail	B	3	2	1	B	3	2	1
Avg. Comp. Time	921.1	1105.3	1119.8	1136.8	1922.7	2629.0	2673.6	2696.7
% Comp. Ops	93.3	92.0	91.9	91.7	85.1	79.2	78.8	78.6

	30% Sybil Nodes				**40% Sybil Nodes**			
Max Fail	B	3	2	1	B	3	2	1
Avg. Comp. Time	3463.5	4742.5	4783.4	4761.6	4475.1	6081.0	6094.9	6124.7
% Comp. Ops	72.1	61.4	61.0	61.1	63.5	50.1	49.9	49.7

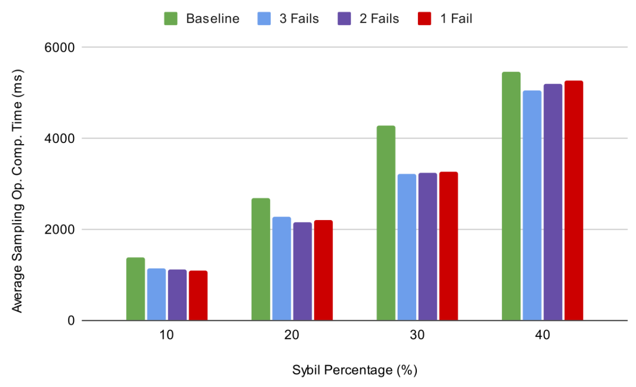

Fig. 3. Average Completion Time for a 300 Node Network

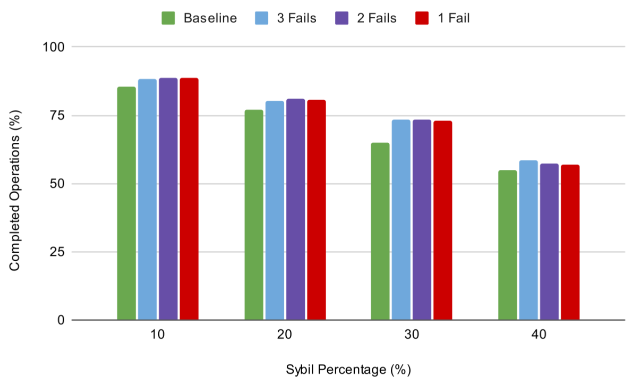

Fig. 4. Average Percentage of Completed Operations for a 300 Node Network

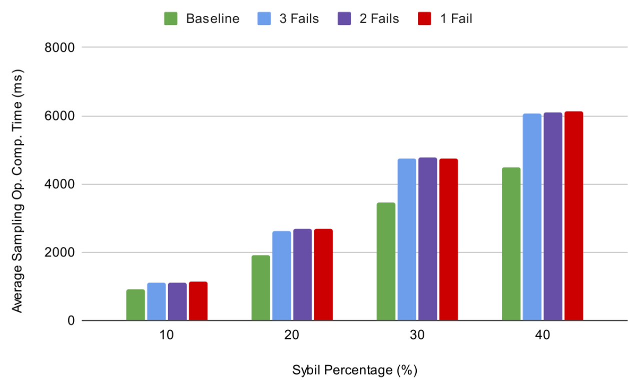

Fig. 5. Average Completion Time for a 2000 Node Network

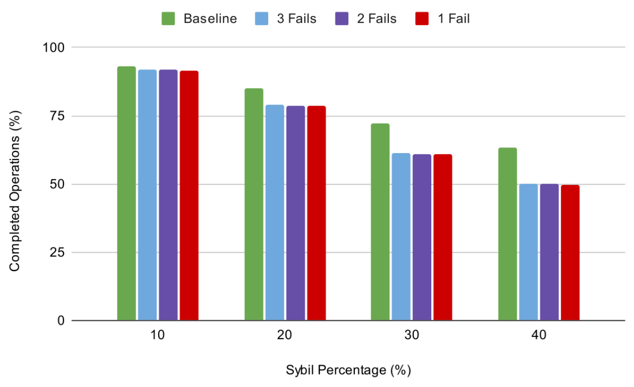

Fig. 6. Average Percentage of Completed Operations for a 2000 Node Network

VII. Discussion

The results show that the mechanism can improve the average time taken to complete a sampling operation and the proportion of positive sampling operations in smaller network sizes rather than larger ones. Given a network size of 300 Nodes with a population of 30% sybils, the mechanism improves the network's performance such that under such conditions, the network can now meet the 4-second consensus deadline imposed by the DAS process where it otherwise could not.

Up to 40 % sybil nodes in a 300-node network, the network still shows some albeit negligible resilience against large proportions of sybil nodes - however, this demonstrates a promising trend in the mechanism's overall performance.

The results show that the network performance drops overall as the percentage of sybils increases. This is expected behaviour because the larger the proportion of sybils conducting a data withholding attack, the fewer legitimate nodes there are to conduct normal network operations. As the percentage of sybils increases, it takes longer for a sampling operation to complete on average and the proportion of successfully completed sampling operations drops.

The results differ significantly as the network's size increases. Given the 16 "child" limit placed upon each "parent," it is likely that the structure of the Rated List changes significantly as the network's size increases. Under larger network sizes, this may undermine the mechanism's parental structure and appears to affect legitimate nodes disproportionately.

As the number of malicious nodes increases under larger network sizes, the performance of a network using the mechanism is worse or equal to that of a network utilising no mechanism. Further research is necessary to understand how the Rated List is constructed in such cases and how to implement sybil countermeasures effectively.

One primary drawback of the current Rated List mechanism is the differing implementation of the "Root" children. The total number of children the Root can hold in the current implementation is infinite. This acts as a measure to allow previously unseen (parentless) nodes access to be rated in the Rated List and, therefore, to be considered for sampling (albeit as a node with zero initial reputation). In some cases, it may be possible for this layer to become overpopulated because it is of infinite length, which undermines the parental structure implemented by the Rated List. Further research is necessary to understand how previously unseen nodes can be treated to support the mechanism effectively.

Previously unseen parentless nodes currently exist in the kademlia-simulator mechanism, but do not exist in real life. More work is necessary to understand how the Rated List needs to interact with these nodes and to investigate whether the Rated List implementation cannot find the parents of these nodes due to malfunction.

The results also show that the maximum tolerated failures for optimal results varies depending on the proportion of malicious nodes. For example, in a network of 300 nodes where 10% are sybils, one maximum tolerated failure will give the highest performance. In contrast, if 40% are sybils, three maximum tolerated failures will give the highest performance.

This result shows that in a case where nodes are less likely to be sybils, a stricter approach to pruning them from search tables is more effective. This may be because the attack's overall impact on the network is lower. This enables a validator to prune perceived "bad" nodes immediately, as there are likely to be many nodes that still have the required cell available for sampling.

VIII. Conclusion

This project has implemented and evaluated the performance of the Rated List security mechanism in the DAS process in DHTs.

The project has shown that the implemented Rated List mechanism improves the performance of a network with sybil nodes, even in severe circumstances with more than 40% sybils, and can greatly improve the network's performance with up to 30% sybils.

While the project has shown promising improvements in some cases, it has also highlighted the difficulties of implementing a one-size-fits-all scalable security mechanism for networks of different sizes. Further work is needed to understand how the structure of the Rated List can be modified to scale appropriately with the size of the network.

Despite this, the Rated List's performance represents a promising starting point for investigating more effective sybil-resistant mechanisms implemented as an extension to the existing DAS mechanism. The proposed mechanism's improvements have shown that scalable solutions to reliable, secure and scalable data availability verification are feasible and could be practical for safe future implementation and adoption.

A. Research Questions

As per **RQ1**, the proposed mechanism does improve DHT robustness to sybil attacks in cases where the number of nodes is proportional to the structure of the Rated List. Further investigation is required to understand why this is the case and what can be done to remedy it.

As per **RQ1.1**, a DHT using the proposed mechanism can recover from a large-scale sybil attack in the same validation period in which the attack occurs, given specific performance parameters. Firstly, the size of the network must be compatible with the size of the Rated List, and second, the proportion of sybils must be within a certain bound. In this project's case, for 300 nodes, the proportion of sybil nodes must be around 30%, at which point the mechanism can prevent an attack from delaying a validation operation after the 4000 ms cutoff point.

B. Future Work

Many potential improvements can be made to the mechanism as it exists now. Firstly, performance must be tuned to calculate how the size of the Rated List itself can be tuned to the total size of the network—currently, the static size of the Rated List is very likely to have affected performance.

In addition, it is possible to investigate how additional potential countermeasures can be implemented to improve the performance of the Rated List in circumstances where there is a larger proportion of sybils - although it is valuable to note that once the number of sybils exceeds the number of legitimate nodes, the consensus of the network is already compromised.

Potential future countermeasures include diverse parent sampling - discussed in Section IV-C - where the Rated List prioritises nodes with the most diverse set of parents to avoid nodes with common sybil or malicious parents. While a naive implementation such as the one attempted in this paper did not help, the parental structure of the Rated List has the potential for many additional nuanced approaches to be implemented. In addition, one approach theorised by Feist involves subtree pruning, where the entire subtree of sybils is cut off at the "root" of the attack. This may be difficult to implement given the infinite root node width used by the current Rated List implementation. However, the existing mechanism provides a promising basis for this approach.

C. Ethical Considerations

The functionality implemented in this project is intended solely as a contribution to the public kademlia-simulator Java project authored by Datahop and as a proof-of-concept showing that the Rated List's mechanism can improve DAS. The mechanism's performance was tested with synthetic data within an isolated environment. The mechanism would be subject to extensive and detailed review before being implemented in any real-life environment.

IX. Acronyms

TPS	Transactions Per Second
DAS	Data Availability Sampling
DHT	Distributed Hash Table

References

[1] Alexandros Antonov and Spyros Voulgaris. "SecureCyclon: Dependable Peer Sampling". en. In: *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. Hong Kong, Hong Kong: IEEE, July 2023, pp. 1–12. ISBN: 9798350339864. DOI: 10.1109/ICDCS57875.2023.00041. URL: https://ieeexplore.ieee.org/document/10272533/ (visited on 06/07/2024).

[2] Ingmar Baumgart and Sebastian Mies. "S/Kademlia: A practicable approach towards secure key-based routing". en. In: *2007 International Conference on Parallel and Distributed Systems*. Hsinchu, Taiwan: IEEE, 2007, pp. 1–8. ISBN: 978-1-4244-1889-3. DOI: 10.1109/ICPADS.2007.4447808. URL: http://ieeexplore.ieee.org/document/4447808/ (visited on 06/05/2024).

[3] George Danezis et al. "Sybil-Resistant DHT Routing". en. In: *Computer Security – ESORICS 2005*. Ed. by David Hutchison et al. Vol. 3679. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 305–318. ISBN: 978-3-540-28963-0 978-3-540-31981-8. DOI: 10.1007/11555827_18. URL: http://link.springer.com/10.1007/11555827_18 (visited on 07/04/2024).

[4] John R. Douceur. "The Sybil Attack". en. In: *Peer-to-Peer Systems*. Ed. by Peter Druschel, Frans Kaashoek, and Antony Rowstron. Berlin, Heidelberg: Springer, 2002, pp. 251–260. ISBN: 978-3-540-45748-0. DOI: 10.1007/3-540-45748-8_24.

[5] Michał Król et al. *Data Availability Sampling in Ethereum: Analysis of P2P Networking Requirements*. en. arXiv:2306.11456 [cs]. June 2023. URL: http://arxiv.org/abs/2306.11456 (visited on 06/05/2024).

[6] Petar Maymounkov and David Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". en. In: *Peer-to-Peer Systems*. Ed. by Gerhard Goos et al. Vol. 2429. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65. ISBN: 978-3-540-44179-3 978-3-540-45748-0. DOI: 10.1007/3-540-45748-8_5. URL: http://link.springer.com/10.1007/3-540-45748-8_5 (visited on 06/03/2024).

[7] Riccardo Pecori. "S-Kademlia: A trust and reputation method to mitigate a Sybil attack in Kademlia". en. In: *Computer Networks* 94 (Jan. 2016), pp. 205–218. ISSN: 13891286. DOI: 10.1016/j.comnet.2015.11.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S1389128615004168 (visited on 06/04/2024).

[8] Antony Rowstron and Peter Druschel. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". en. In: *Middleware 2001*. Ed. by Rachid Guerraoui. Berlin, Heidelberg: Springer, 2001, pp. 329–350. ISBN: 978-3-540-45518-9. DOI: 10.1007/3-540-45518-3_18.

[9] I. Stoica et al. "Chord: a scalable peer-to-peer lookup protocol for Internet applications". In: *IEEE/ACM Transactions on Networking* 11.1 (Feb. 2003). Conference Name: IEEE/ACM Transactions on Networking, pp. 17–32. ISSN: 1558-2566. DOI: 10.1109/TNET.2002.808407. URL: https://ieeexplore.ieee.org/abstract/document/1180543?casa_token=AlXijjndm0cAAAAA:XvBPExpCqSvMV0DElJWbR4cUdM-012Y-2vXd6y_cCHD1_cjxUpTue8MUbnG0SxSVRyhhvzW3fKf8 (visited on 07/04/2024).

[10] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. "Blockchain Scaling Using Rollups: A Comprehensive Survey". In: *IEEE Access* PP (Aug. 2022). DOI: 10.1109/ACCESS.2022.3200051.

[11] Jyoti Yadav and Ranjana Shevkar. "Performance-Based Analysis of Blockchain Scalability Metric". en. In: *Tehnički glasnik* 15.1 (Mar. 2021). Publisher: Sveučilište Sjever, pp. 133–142. ISSN: 1846-6168, 1848-5588. DOI: 10.31803/tg-20210205103310. URL: https://hrcak.srce.hr/253034 (visited on 07/05/2024).

[12] Mingchao Yu et al. "Coded Merkle Tree: Solving Data Availability Attacks in Blockchains". en. In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Cham: Springer International Publishing, 2020, pp. 114–134. ISBN: 978-3-030-51280-4. DOI: 10.1007/978-3-030-51280-4_8.