Active Sybil Attack and Efficient Defense Strategy in IPFS DHT

Victor Henrique de Moura Netto, Thibault Cholez and Claudia-Lavinia Ignat
Université de Lorraine, CNRS, Inria, LORIA
Nancy, France
{victor-henrique.de-moura-netto, thibault.cholez, claudia.ignat}@inria.fr

***Abstract*—The InterPlanetary File System (IPFS) is a decentralized peer-to-peer (P2P) storage that relies on Kademlia, a Distributed Hash Table (DHT) structure commonly used in P2P systems for its proved scalability. However, DHTs are known to be vulnerable to Sybil attacks, in which a single entity controls multiple malicious nodes. Recent studies have shown that IPFS is affected by a passive content eclipse attack, leveraging Sybils, in which adversarial nodes hide received indexed information from other peers, making the content appear unavailable. Fortunately, the latest mitigation strategy coupling an attack detection based on statistical tests and a wider publication strategy upon detection was able to circumvent it.**

In this work, we present a new active attack, with malicious nodes responding with semantically correct but intentionally false data, exploiting both an optimized placement of Sybils to stay below the detection threshold and an early trigger of the content discovery termination in Kubo, the main IPFS implementation. Our attack achieves to completely eclipse content on the latest Kubo release. When evaluated against the most recent known mitigation, it successfully denies access to the target content in approximately 80% of lookup attempts.

To address this vulnerability, we propose a new mitigation called SR-DHT-Store, which enables efficient, Sybil-resistant content publication without relying on attack detection but instead on a systematic and precise use of region-based queries, defined by a dynamically computed XOR distance to the target ID. SR-DHT-Store can be combined with other defense mechanisms resulting in a defense strategy that completely mitigates both passive and active Sybil attacks at a lower overhead, while allowing an incremental deployment.

I. Introduction

Since the early 2000s, peer-to-peer (P2P) networks have been known to be vulnerable to Sybil Attacks [1], in which an attacker instantiates many fake peers under their control to disrupt the network's operations. Structured P2P networks based on Distributed Hash Tables (DHTs), such as Kademlia [2], are particularly threatened as each peer is responsible for indexing a range of content space according to its position in the DHT. This vulnerability leads to localized attacks, where an attacker forges node IDs to insert multiple Sybils into a specific region of the DHT, gaining control over the indexing mechanism and denying access to targeted content [3], [4]. To counter such threats, several defense mechanisms have been proposed [5], [6], typically relying on evaluating the distribution of the peers around an ID using statistical tests of relative entropy, to detect anomalies indicative of Sybil activity. Additionally, they may modify the lookup strategy to either avoid suspicious nodes or reduce their influence by diluting their weight in the search process.

In recent years, the InterPlanetary File System (IPFS) [7] has emerged as an attempt to build the storage layer for the decentralized World Wide Web (WWW). IPFS is a suite of protocols designed for data exchange, built on the principles of content addressing and peer-to-peer networking. Data is never directly uploaded to the network, only references pointing to the providers of the content, who are contacted directly for data exchange. The primary implementation of IPFS is Kubo [8] (formerly known as go-ipfs), written in Go. IPFS is used in a variety of applications, including Berty [9], a peer-to-peer messaging app that operates without central servers; DTube [10], a decentralized video-sharing platform; and Filecoin [11], a cryptocurrency-based decentralized file storage network. The core networking functionality of IPFS has been extracted into a separate project called libp2p [12], an open-source, modular framework for building P2P applications.

Recent studies [6], [13] have shown that the main IPFS client has so far neglected to implement security mechanisms against Sybil attacks, and was consequently highly affected by a basic Sybil attack scenario. In such attack, an attacker can eclipse content from the network by precisely inserting Sybil nodes with forged IDs. To address this issue, the authors of [6] proposed a detection and mitigation strategy that reduces the impact of Sybil nodes. Their approach involves extending the set of peers that receive publication records by performing region-based queries, based on their common prefix length (CPL) with the CID, thereby defeating content censorship attempts.

In this paper, we propose a new attacker model that optimizes both the number and position of Sybils while remaining below the detection threshold of the proposed countermeasure. As a result, we show that the majority of publication records can still be intercepted by Sybils without triggering the defense mechanism. We exploit this vulnerability by introducing active behavior in Sybils, which now advertise fake records to trigger early termination in IPFS's lookup process, leading to an attack success rate of 80%. In response, we propose a new Sybil-resistant strategy for storing data in the DHT. This approach relies on an efficient way for calculating the XOR distance where the peers responsible for a record should be located, combined with a systematic use of region-based queries that we have optimized to significantly reduce overhead. Our

mechanism called SR-DHT-Store, when used in combination with other mechanisms such as disjoint lookup paths [14], fully mitigates the active attack and defines a new best practice for securely publishing data in a DHT.

To summarize, our main contributions are the following:

- a combinatorial optimization algorithm for inserting Sybil nodes while remaining undetected against [5] or [6];
- an active attacker model that exploits a vulnerability to prematurely abort the content lookup process;
- an efficient method for estimating the proper distance of peers responsible for content, based on the XOR metric rather than the CPL;
- a systematic use of region-based queries with a reduced overhead;
- an evaluation of combined defense mechanisms against Sybil attacks with our proposed mitigation strategy.

The remainder of the paper is organized as follows. Section II and Section III present the technical background on IPFS and related work on Sybil attacks and defense mechanisms for DHTs, with a specific focus within the context of IPFS. Section IV presents and evaluates our active Sybil attack scenario and its results. Our Sybil-resistant publication strategy for the DHT, SR-DHT-Store, is described and evaluated in Section V. Finally, in Section VI, we further enhance DHT resilience by combining our proposed publication strategy with additional mechanisms, in particular S/Kademlia, before concluding the paper in Section VII.

II. Background on IPFS

In this section, we provide a comprehensive overview of the key concepts necessary to understand this work. We begin by introducing peer and content identifiers, followed by a discussion of the principles of the DHT. As a core component of IPFS, we then examine the DHT's internal structures, such as the Routing Table (RT), along with the mechanisms used for content addressing and retrieval.

A. Identifiers

Each piece of content in the network is assigned a unique identifier, known as a Content Identifier (CID), derived from the hash of the file's contents. This hash is computed from a Merkle Directed Acyclic Graph (MerkleDAG) [15], which splits the content into fixed-size segments, called chunks, and organizes them into a tree-like structure. The root of this tree, is obtained from the hash of all the chunks and serves as the CID for the content. As a result, any modification to the content produces an entirely different CID, allowing recipients to verify the authenticity and integrity of the data by recalculating the hash of the received content. The root hash is encoded using the Multiformats standard [16], producing a self-describing, future-proof value that supports multiple hashing algorithms, encoding schemes, and serialization formats.

Peers are identified by Peer Identifiers (PID), which are derived from the hash of a locally generated public keys. This PID is formatted similarly to CIDs, using the Multiformats standard. To establish a secure connection, peers exchange self-signed certificates that include their public key, along with a signature generated using their respective private key, as an extension to the certificate. This ensures the authenticity of both the source and destination peers while establishing a reliable connection for secure data exchange.

B. DHT

Nodes and content become accessible once they are inserted into the DHT, which operates within a 256-bit key space. This distributed hash table contains pairs of keys and values, distributed along multiple nodes in the network. The keys are the SHA-2 hash of a CID or PID, ensuring a fixed-size, comparable value regardless of their original format. There are two main elements inserted into this DHT: provider records (PR) and peer records.

When providing content to the network, a **provider record**, containing the pair {*key:* provided CID, *value:* provider PID}, is distributed to selected peers. These peers store this record in their local hash tables, which are periodically cleaned to avoid retaining references to unused content due to network churn. By default, to ensure the continuous availability of the content, the provider must periodically announce the record, a process known as content pinning [17].

After identifying the content provider from an obtained PR, the node must retrieve the provider's IP address to initiate the content exchange. This is done by searching for the corresponding **peer record**, which maps {*key:* PID, *value:* multiaddresses}. The values of those pairs are multiaddresses [18], a standardized format used to represent a peer's contact information, including all supported protocols, IP addresses, and open ports.

To searching and providing records in the DHT, nodes perform a **DHT lookup**, an iterative process to find the k closest nodes to a target identifier. Initially, the α closest nodes to the target on its routing table are queried in parallel for the closest nodes they know to the identifier. All responses are stored in a shared data structure, ordered by distance, and used in subsequent lookup iterations. The process continues until the closest β nodes found have already been queried and have successfully responded. IPFS follows the Kademlia protocol specifications [2], using the XOR metric to measure distance between identifiers, and the parameters: $k = 20$, $\alpha = 10$, and $\beta = 3$ [19].

C. Routing Table

When joining the public network, a bootstrap process begins by contacting one of the official IPFS bootstrap nodes. They serve as entry points for populating the node's routing table and announcing their presence to the network. The RT consists of i-buckets, where $i \in \{0, 255\}$, each capable of storing up to k nodes that share exactly i leading common bits with the node's own identifier. In this paper, i is also referred to as the common prefix length (CPL) relative to another identifier.

To keep all buckets updated, the routing table is refreshed every ten minutes, as well as immediately after the bootstrap process. During each refresh, the node removes peers that are

unreachable, offline, or have not proven useful within a defined time window [20]. Each bucket $i \in \{0, 255\}$ is updated by performing a lookup toward a randomly generated identifier that shares exactly i common prefix bits with the node's PID. Peers discovered during this process are organized into the appropriate buckets, based on the CPL they share with the node's PID. In practice, IPFS performs this operation only for the first $0 \leq i \leq 15$ buckets. This update is then followed by a lookup toward the node's own PID to refresh the remaining $i \geq 16$ buckets.

D. Content Addressing and Retrieval

To make content available on the network, provider records are distributed to the k nodes closest to the precomputed CID. These nodes are obtained through a DHT lookup, and each of the k nodes is contacted individually to receive the record, becoming *resolvers* for the content. This process is illustrated in Figure 1(a).

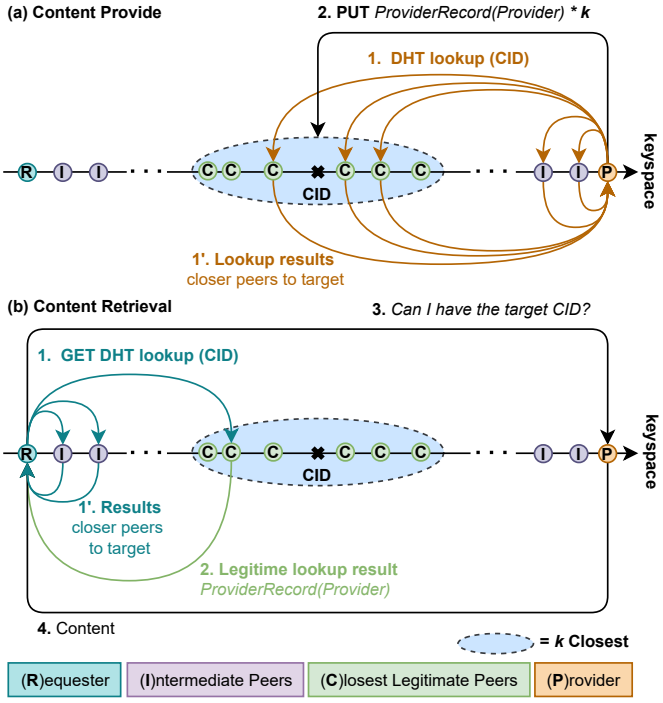

Fig. 1: Content provide/retrieval in IPFS DHT. This example assumes $k = 6$.

Content retrieval in IPFS is divided into three main steps. First, each peer in the Swarm is individually queried for the requested content, checking whether they have a copy of the desired content. If none of the connected peers possesses the content, the process falls to the DHT. The retrieval DHT lookup is slightly modified compared to a standard Kademlia lookup: it opportunistically requests both the provider record of the content and the k closest peers known to each queried node. The lookup continues until a valid provider is found, and the content is retrieved, or until ten provider records have been collected, a termination further exploited in this paper to perform an active attack. All data exchanges in IPFS are made using Bitswap [21]. This retrieval process is illustrated in Figure 1(b).

III. Related Work

In open and fully distributed P2P networks such as IPFS, there is no centralized authority, no privileged peers, and no mechanism to verify the association between a peer identifier and a specific user or device. This lack of centralization enables a single entity to assume multiple malicious identities, known as **Sybils** [1]. By strategically positioning these Sybil nodes, an attacker can gain control over the distributed network and carry out either passive or active attacks, by eclipsing content or nodes from the network. In this section, we review past Sybil-based attacks and corresponding countermeasures applied to Kademlia DHT and IPFS.

A. DHT Attacks

Eclipse attacks, first described by Singh *et al.* [22], aim to isolate a target node or content from the network by surrounding it with malicious peers, effectively hiding it from the rest of the system. In this section, we focus on studies that explore two main strategies for executing this attack: manipulating the routing table and disrupting internal DHT operations.

Initially focusing on routing table manipulation, Wang *et al.* [23] implemented a targeted routing attack against the Kad network, which is based on the Kademlia DHT. They introduced a method for hijacking existing connections in a peer's routing table by impersonating legitimate nodes. By querying victims and mapping their routing tables, the attacker sends spoofed messages to update the contact information of the nodes listed in the RT. As a result, the victim becomes dependent on malicious nodes to communicate with the rest of the network. This attack exploits Kad's lack of identifier verification during peer exchanges, an issue addressed in IPFS.

Another variation of the eclipse attack involves exploiting the interval structure of the DHT. Steiner *et al.* [3] describe an attack in which adversaries eclipse content by inserting malicious nodes near the target identifier using the XOR metric. Since these malicious nodes become the only resolvers for the content, they can control how they respond to queries, effectively dictating access to the targeted data. Cholez *et al.* [4] demonstrated that controlling content retrieval can be achieved using just k Sybil nodes by efficiently eclipsing each of the k file records sent during a publication. Wang *et al.* [23] proposed a more active strategy, where attackers respond to lookup requests with an excessive number of bogus keywords. This triggers early termination due to a keyword match limit, preventing legitimate queries from completing. In the work of Kohnen *et al.* [24], attackers respond to lookup requests with Sybil identities that are only slightly closer to the target than the current best-known nodes. This causes the lookup process to make minimal progress and continue indefinitely until it times out, at which point the content is considered unreachable.

B. IPFS Attacks

Beyond IPFS' vulnerabilities inherited from Kademlia, we provide in this subsection an overview of the attacks that have been documented specifically against the IPFS network.

Network Partitioning. The first documented attack on IPFS was presented by Prünster *et al.* [25]. The authors described a method to eclipse a victim node by fully populating its routing table with Sybil nodes, making it dependent on malicious peers to contact the rest of the network. During DHT lookups, the victim sends queries to the closest nodes it knows from to the target identifier, now all Sybils, allowing the attacker to drop or filter requests. Once eclipsed, the victim's only remaining connections to the legitimate network are through its peer Swarm.

They evaluated two strategies: a naïve approach and a more effective one. In the naïve approach, the attackers continuously ping the victim's first $\log_2(n)$ buckets until all reliable entries in the victim's routing table are dropped and replaced with Sybil nodes. In the more effective technique, Sybil nodes gain priority over honest peers by sending unsolicited data to inflate their usefulness scores. As a result, the victim's connection manager evicts legitimate peers in favor of the seemingly more useful malicious nodes.

Although this peer eclipse attack proved effective, it was mitigated in the go-ipfs (formerly Kubo) 0.7 update [26], as further explained in Section III-C.

Content Eclipse. The content eclipse attack was demonstrated by Sridhar *et al.* [6] and Cholez *et al.* [13]. The attacker begins by brute-forcing at least k peer identifiers to generate and instantiate nodes that are closer to the target content than any legitimate peer. During the next republication interval, the provider node unknowingly sends the provider record exclusively to these malicious nodes, which are now the k closest peers to the content. When the Sybils are queried during a DHT lookup, they respond with empty results, claiming no knowledge of the received records. Once the expiration time is reached for the reliable nodes that had previously stored the records, before the Sybil nodes were instantiated, the content becomes eclipsed. This attack is illustrated in Figure 2. For simplicity, the DHT lookup performed by the provider to locate the k closest nodes has been omitted from the figure.

In Kubo, provider records are republished at an interval of $t_r = 22$ hours and expire after a duration of $t_e = 48$ hours. While generating appropriate PeerIDs is computationally intensive due to the large 256-bit keyspace, both studies demonstrated that it is computationally feasible. Cholez *et al.* [13] showed that generating 20 Sybil PeerIDs sufficiently close to any target CID would take less than 1.5 hours on a standard 8-core desktop machine.

C. IPFS Defense Mechanisms

Over the years, various strategies have been proposed to mitigate Sybil attacks. Douceur [1] argues that no distributed system can be fully resistant to Sybil attacks without relying on a centralized certificate authority. However, in a decentralized, open-source protocol like IPFS, a certificate authority is

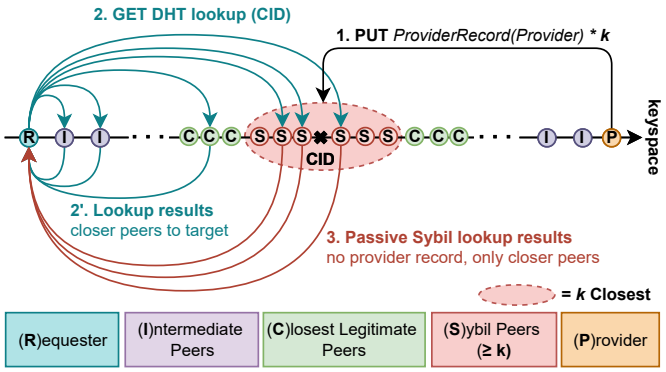

Fig. 2: Content eclipse attack in IPFS DHT. This example assumes $k = 6$.

incompatible. As a result, implementing mitigation strategies is essential to minimizing the impact of Sybil nodes on the network. This subsection presents proposed methods to detect and mitigate Sybil attacks and concludes with an overview of the defenses currently integrated into Kubo.

S/Kademlia. According to the original IPFS paper [7], the protocol should implement two key recommendations from S/Kademlia [14] to improve DHT robustness: identity generation from key pairs and disjoint lookup paths. The first ensures that each node derives its identifier from a local key pair, which is used for identity and message authentication. The second proposes that lookups must be performed along separate, non-overlapping paths, storing responses in isolated queues to reduce the impact of adversarial peers during routing.

However, as explained by Cholez *et al.* [13], Kubo does not enforce the disjoint lookup paths. While it performs parallel queries during lookups, all discovered nodes and provider records are aggregated into a single structure, making the process more vulnerable to Sybil attacks. This limitation is further analyzed in our work.

Attack Detection using the K-L Divergence. Cholez *et al.* [5] propose a Sybil attack detection mechanism based on analyzing the distribution of the k closest nodes to a given target. In a reliable network, nodes are expected to be randomly positioned, resulting in a uniform distribution of identifiers across the DHT space. By estimating the total number of nodes in the network, it is possible to compute the expected model distribution of the k closest nodes to a given identifier. However, when malicious nodes are strategically positioned to occupy all k closest positions, the observed distribution deviates from the expected random distribution. To quantify this deviation, the authors use the Kullback-Leibler (K-L) divergence, which measures the discrepancy between two probability distributions: the expected model distribution and the observed real distribution. The minimum value of the K-L divergence is $D_{KL} = 0$, which indicates no difference between the distributions, increasing as the discrepancy between the distributions grows.

Sridhar *et al.* [6] implemented this attack detection method,

however introducing a variable network size N. To calculate their model distribution, they estimate the probability that a randomly selected closest peer j has a CPL exactly equal to x. This probability distribution, $p(x)$, is then compared with the empirical distribution $q(x)$, which is obtained from the actual k closest nodes to the target content. Both distributions are compared using the K-L divergence equation, as follows:

$$D_{KL} \triangleq \sum_{x \in X} q(x) \ln \left(\frac{q(x)}{p(x)} \right) \quad (1)$$

Through experiments on the IPFS network, they established a threshold value of 0.94. If the discrepancy between the observed and expected distributions exceeds this threshold, the distribution is classified as an attack. The authors prioritized minimizing false negatives to increase the likelihood of detecting an attack when defining this constant, at the cost of introducing some overhead. Their threshold is calibrated for a false positive rate of 4.4% and a false negative rate of 0.81%.

Region-Based Queries. Sridhar *et al.* [6] additionally proposed a mitigation strategy to be applied upon attack detection. Their approach is based on the assumption that the k closest reliable nodes cannot be removed from the network by an adversary. To reach these reliable nodes, they introduce the concept of region-based queries. Using the previously estimated network size from the attack detection phase, they calculate the *CPL* at which the k-th closest reliable node would be positioned in a truly random distribution of identifiers within the network space. Lookups are then performed until all nodes that share at least this *CPL* with the target are identified, afterward sending PRs to each of them. As a result, regardless of the number of Sybils deployed, approximately $k = 20$ reliable nodes receives the provider record, significantly increasing the likelihood of a requester successfully retrieving the content. This mitigation strategy is illustrated in Figure 3.

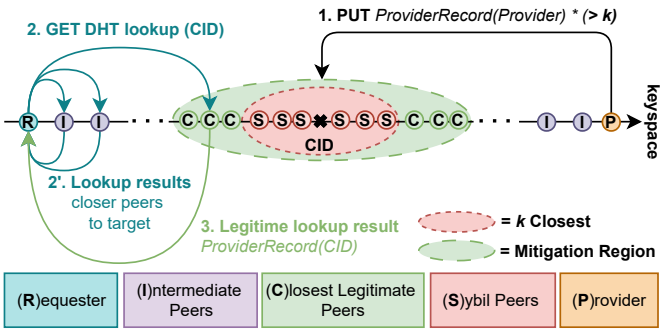

Fig. 3: Region-based query mitigation strategy. This example assumes $k = 6$.

The k-th closest reliable node in a random distribution should contain a *CPL* of $minCPL = \lceil \log_2(\frac{N}{k}) \rceil$, delimiting the region containing a theoretical region of k reliable nodes. Their search algorithm begins with an initial DHT lookup to identify the k closest nodes to the target CID. The k-th closest node returned from the lookup shares a given *CPL* with the target, meaning that all other returned nodes share at least the same *CPL* with the target CID. In the subsequent iteration, a random identifier with $CPL+1$ with the target is generated, to locate nodes farther from the target through additional lookups. This process is repeated until the *CPL* of the k-th closest node is less than *minCPL*, ensuring that all nodes within the *minCPL* region have been successfully identified.

At the time of writing, this mitigation has not been implemented in IPFS to address content eclipse attacks. In the following sections of the paper, we occasionally use the term "region-based queries" to refer collectively to both the attack detection mechanism and this associated mitigation strategy.

Kubo Constraints. The go-ipfs 0.7 release [26] addressed the Prunster *et al.* attack [25]. In this version, IPFS introduced a mechanism that ensures active connections in the routing table are pruned only during idle periods, preserving long-lived, stable peers. Additionally, a diversity filter was implemented [27], limiting the addition of peers from the same /16 IPv4 subnetwork to a maximum of three in the routing table, and no more than two per bucket. However, this restriction applies only to the routing table and does not affect the lookup process, which remains vulnerable to encountering multiple peers with the same IP address, as highlighted in [13].

A more recent addition in version 0.34.0 of go-libp2p [28] introduces a limitation of a maximum of eight concurrent connections per /16 subnetwork. This restriction prevents a peer from maintaining an excessive number of connections within the peer Swarm, complementing the existing diversity filter, which already imposes stricter limits on entries in the routing table.

While the passive content eclipse attack is likely to be mitigated in IPFS, no study has yet evaluated the network's resilience to an active content eclipse attack. An active attack involves malicious nodes actively responding to requests with semantically correct but falsified data. The following section investigates the impact of such attack on the IPFS network, with a particular focus on defeating the latest defense mechanism proposed for mitigating Sybil-based content censorship in IPFS [6].

IV. Active Sybil Attack on IPFS

This section begins by introducing an active attacker model that targets the IPFS network by exploiting a premature termination condition in the current DHT lookup process. The model is then adapted to specifically exploit the attack detection and region-based query mechanisms proposed in the most recent defense strategy for the network [6]. Finally, we analyze the cost associated with executing the attack and evaluate its effectiveness.

A. Attacker Model

Our attack exploits an early termination mechanism by responding to lookup requests with a falsified list of provider records when queried about the targeted content. During a DHT lookup for a content's provider record, the query is designed to stop once a predefined maximum number of records is retrieved. In Kubo, this limit is set to ten records, which can be exploited

by a single or multiple malicious nodes positioned along the lookup path. Those nodes can respond with semantically correct but false records, causing the lookup to terminate prematurely.

The active approach achieves the same success rate as a passive attack when at least k malicious nodes are instantiated around the target, as all k provider records are exclusively sent to these adversarial nodes. As previously demonstrated, when k Sybil nodes surround the target, the content can be completely eclipsed from the DHT. However, by exploiting the premature termination of the DHT lookup process, the attacker can reduce the number of Sybil nodes required to execute the attack. If the adversary's nodes consistently manage to be the first contacted during the victim's lookup, the content can be effectively eclipsed from the network with as few as $n \leq k$ malicious peers.

Since no mitigation has been implemented against the passive attack in the current Kubo version 0.33.0, presenting results using the active approach on this last release is uninformative, as the attack will always succeed. Therefore, our work focuses on implementing[1] the active attack against the most effective mitigation proposed for IPFS to counter content eclipse: the attack detection and region-based queries *et al.* [6].

Termination Exploit. The lookup process, implemented in the function `findProvidersAsyncRoutine`[2], declared in the file `routing.go`, includes a call to `runLookupWithFollowup`, defined in `query.go`, which contains three distinct search termination conditions. One of these terminations is an anonymous function passed as a parameter to the variable `stopFn`, at line 351 of `routing.go`, and shown in Listing 1. In this function, `psSize` represents the current number of provider records found during the lookup, while `count` denotes the maximum number of accepted records, which is set to ten in Kubo. If the number of retrieved records exceeds `count`, the lookup process is terminated. This specific termination is referred to as an external stop. Once triggered, it initiates a continuous lookup process that is repeated every minute until the request is either successfully completed or canceled.

```
1   func() bool {
2       return !findAll && psSize() >= count
3   },
```

Listing 1: The function `stopFn` exploited in IPFS to achieve early termination of the DHT lookup.

Malicious nodes can respond to queries with records containing randomly generated identifiers and IP addresses. Since the peer cannot establish contact with these nodes using the fake addresses, it attempts to discover their potential new contact information by performing additional DHT lookups.

Strategy. Our strategy against their mitigation involves strategically inserting the maximum number of Sybil nodes while maintaining the appearance of a randomly generated peer distribution. The attacker begins by computing the expected model distribution, similarly to the attack detection mechanism, assuming an ideal scenario where all identifiers are randomly distributed across the DHT. Next, the attacker retrieves the k closest nodes to the target content and calculates the K-L divergence between the observed and model distributions. In the absence of a false attack detection, the resulting divergence should remain below the established detection threshold. We exploit the margin left in the threshold, which is needed to maintain the false positive rate at an acceptable level, for inserting Sybil nodes in optimal positions without raising suspicion. The attack strategy is illustrated in Figure 4.

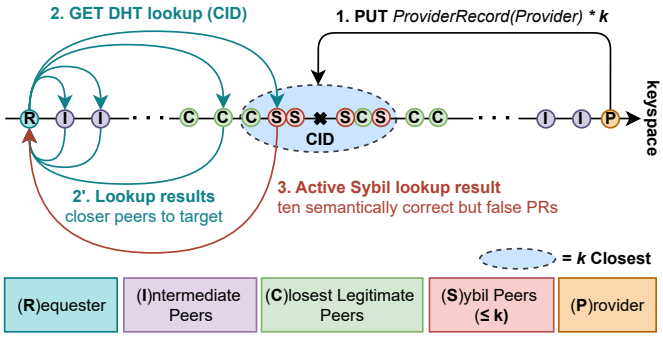

Fig. 4: Active attack against the proposed mitigation. This example assumes $k = 6$.

Average D_{KL}. Around a non-attacked content, the distribution of the k closest nodes in the network should resemble the model distribution, resulting in a low $D_{KL} < threshold$. To assess this, we computed the D_{KL} of the k closest distribution of 100 random identifiers. The model distribution was generated using the network size estimation provided by `go-libp2p-kad-dht`[3], computed prior to each test lookup. This approach avoids relying on a potentially biased network size throughout our experiments. As shown in Figure 5, the majority of D_{KL} values fall below the threshold defined by Sridhar et al., with 83% of the samples below $D_{KL} = 0.7$. This highlights opportunities for strategically placing Sybil nodes in optimal positions while still staying below the detection threshold. On the other hand, it highlights the false positives, which account for 11% of the lookups.

Optimizing Sybil Placement. To determine the optimal positioning of Sybil nodes while minimizing the risk of attack detection, we formulate the problem as a correlated combinatorial optimization, similar to the knapsack problem [29]. In this problem, adding a malicious node to the distribution increases the score or profit p_j, while also increasing the distribution's D_{KL}, which corresponds to the weight w_j of the solution. Our algorithm begins by identifying the k closest nodes to the target CID and organizing them into an array that tracks the current

[1] https://github.com/vicnetto/active-sybil-attack-and-efficient-defense-strategy-in-ipfs-dht.

[2] https://github.com/libp2p/go-libp2p-kad-dht/blob/e676e51718b8fca0d96282523967f8632534f0bd

[3] https://github.com/libp2p/go-libp2p-kad-dht/blob/0af416971f51f07a60afb16851f030deea29ce2b/netsize/netsize.go

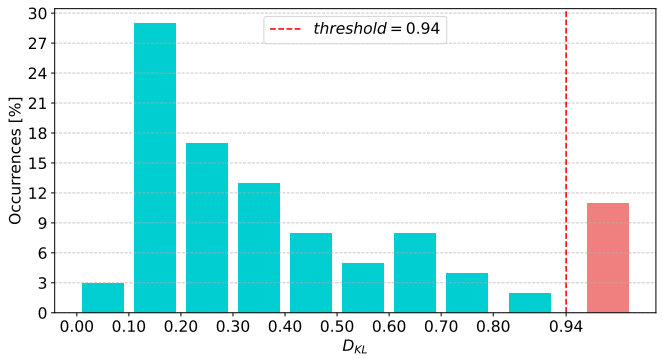

Fig. 5: D_{KL} Distribution over 100 Random DHT Requests.

number of peers per CPL. Using an estimated network size, we then compute the D_{KL} of the current distribution.

Our algorithm is illustrated in Algorithm 1. It contains three parameters: the results, which are empty at the start; the current nodes per CPL; and the current analyzed CPL. This last parameter starts at the CPL where instantiating at least one Sybil node would not immediately trigger attack detection due to its closeness to the target.

For each CPL in the distribution, there are $k+1$ possible node configurations, ranging from 0 (no nodes) to k (all k closest nodes). If n_{reliable} reliable nodes are already present at a given *CPL*, the valid range for Sybil insertion becomes $0 \leq n_{\text{sybil}} \leq k - n_{\text{reliable}}$. Each value within the range of n_{sybil} is individually tested by updating the current distribution using the `updateKClosestWithSybils` function, which attempts to remove the n_{sybil} farthest nodes to make space for this quantity of Sybil nodes. If successful, the function returns the updated distribution, and otherwise, it returns nothing. By evaluating the return of this function, we recalculate the D_{KL} value to verify that the updated distribution still remains below the detection threshold. If the condition is met, the algorithm recursively proceeds to $CPL-1$, until reaching $CPL=0$. At this point, the result is added to the result set for evaluation.

In the attack detection mechanism proposed by Sridhar *et al.*, the comparison is based on the number of nodes per CPL between the model and the observed distribution, without considering the actual proximity of nodes within each CPL according to the XOR distance. However, nodes with the same number of shared prefix bits can still differ significantly in their distance to the target. To exploit this limitation, we introduce a finer level of granularity in our attack strategy by ensuring that optimized Sybil nodes are always positioned closer to the target than reliable nodes that share the same CPL. As a result, any reliable nodes located in the farthest CPLs are removed from the set of the k closest nodes, as we can substitute them with closer Sybil nodes that share the same common prefix length, while maintaining the same D_{KL} value and avoiding detection.

Score. Based on our observations, each valid distribution should be evaluated according to the following three objectives to determine its suitability as an optimal attack distribution: *(i)* maximize the number of Sybil nodes; *(ii)* position the Sybil nodes as close as possible to the target; and *(iii)* ensure that the closest node to the target is a Sybil. These objectives lead to our scoring function:

$$score = \sum_{cpl=0}^{255} s(cpl) \cdot cpl. \tag{2}$$

We iterate through all CPLs in the solution, multiplying the number of Sybil nodes in each CPL, obtained using the s function, by their CPL. The function s compares the initial distribution to the resulting one, identifying the number of additional malicious nodes inserted at each CPL. In case of a score tie, the tie-breaking criterion is the distribution with the lowest D_{KL}.

Since the network size estimation can vary between nodes, the model distribution and consequently the D_{KL} value can also change. To avoid discrepancies between the attacker's and provider's estimations, we enforce a maximum $D_{KL} \leq 0.85$. This threshold accounts for an approximate 10% margin of error in the network size estimation. When the provider underestimates the network size, the attack becomes easier to detect, as the model distribution tolerates fewer close nodes. To evaluate the impact of this variance on detection, we compared an attack generated with $D_{KL} \approx 0.85$ using an average estimation from a 24-hour test period, $ns_{average} = 13239$, against a detection performed using different network size estimations. When tested with the lowest recorded network size from that period $ns_{lowest} = 12347$, the attack remained undetected. These results are illustrated in Figure 6.

Algorithm 1 Recursive function to optimize the Sybil placement in the k closest nodes from a target content.

Require: *results* - Stores valid distributions found
Require: *nodesPerCpl* - Current k closest on the distribution
Require: *currentCpl* - Current CPL

```
1:  procedure OPTIMIZE_SYBIL_PLACEMENT(results,
      nodesPerCpl, currentCpl)
2:      nodesInCpl ← nodesPerCpl[currentCpl]
3:      for i = nodesInCpl to 20 do
4:          sybils ← i − nodesInCpl
5:          updatedNodesPerCpl ←
              updateKClosestWithSybils(nodesPerCpl, sybils)
6:          if updatedNodesPerCpl == null or
              isKLOverThreshold(updatedNodesPerCpl) then
7:              break                    ▷ Invalid distribution.
8:          end if
9:          if currentCpl − 1 < 0 then          ▷ Termination.
10:             results.add(updatedNodesPerCpl)
11:             continue
12:         end if
13:         OPTIMIZE_SYBIL_PLACEMENT(results,
              updatedNodesPerCpl, currentCpl − 1)
14:     end for
15: end procedure
```

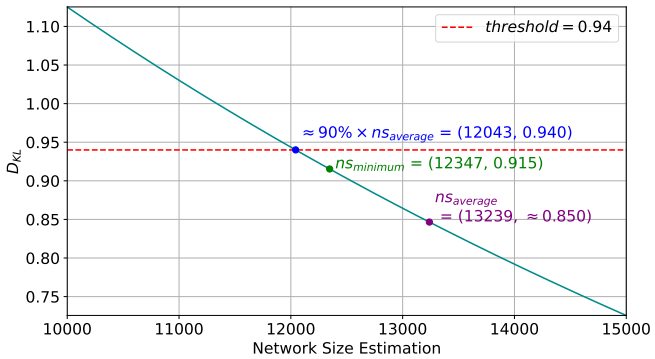

Fig. 6: Impact of network size estimation on an attacked distribution of $D_{KL} = 0.85$.

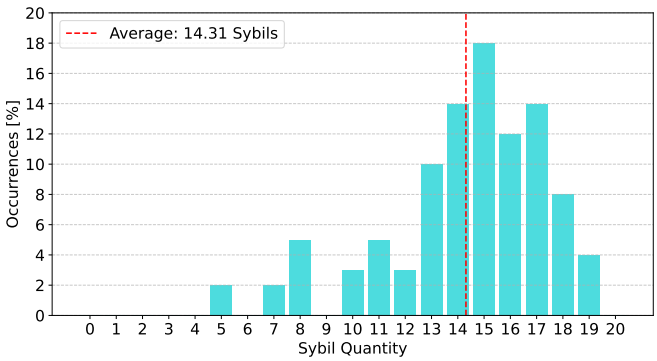

Fig. 7: Optimized Sybil count across 100 random distributions.

B. Cost Analysis

The bandwidth and computing resources needed to instantiate the Sybil nodes are not considered significant, as a publicly accessible IP address and a stable internet connection are sufficient to carry out the attack. The primary cost of executing the active attack is the computational effort required to brute-force Sybil identifiers. While resource-intensive, this process can be completed in a relatively short time.

Brute-force Identifiers. To obtain the malicious nodes, we brute-force identifiers by generating Ed25519 [30] key pairs. The expected number of attempts required to generate a single identifier within a specific CPL is given by $2^{\text{CPL}+1}$. Consequently, the time required to generate a node at a given CPL can be expressed as:

$$t(CPL) = t_{Ed25519} \cdot 2^{CPL+1}. \tag{3}$$

The constant t_{Ed25519} represents the time required to generate an Ed25519 key pair using our implementation. On a machine equipped with a 13$^{\text{th}}$ Gen Intel i7-13700H processor, t_{Ed25519} is approximately $1.35\,\mu s$. Consequently, generating a node with $CPL = 25$ to the target CID would take approximately 1 minute and 30 seconds. While this CPL is considered exceptionally close to the target for a network of size $ns_{average}$, and was not observed in any of our test distributions, such proximity could still be brute-forced in a relatively short time.

C. Experiment Description

Since the number of malicious nodes depends on the distribution of the k closest peers, we first conducted an experiment to determine the average number of Sybil nodes that could be inserted into random peer distributions. As illustrated in Figure 7, the experiment shows an average of 14.31 Sybil nodes among the 20 closest. For this reason, to represent an average-case scenario, all our tests were conducted using distributions that allow the insertion of 14 malicious nodes.

In the same figure, we observe that in at least 91% of the distributions, a minimum of 10 malicious nodes were successfully inserted. During our tests, we did not encounter any distribution where inserting Sybil nodes was impossible. Even better, we noticed that our optimization algorithm cancels falsely detected attacks, transforming any k closest nodes into valid distributions that remain below the detection threshold.

To ensure the reliability of our results, five attacks were executed in parallel against five distinct distributions, involving a total of $14 \times 5 = 70$ Sybil nodes operating simultaneously. The Sybil nodes used in the experiment were modified Kubo implementations, responding to all queries for the target content by providing ten randomly generated provider records. The nodes were deployed on a machine within a public network, ensuring their accessibility for DHT requests. These nodes were interconnected, enabling them to recommend one another during lookup operations and ensure the consistent capture of 14 out of the 20 available records in each content publication.

After the Sybil nodes were instantiated, the CID was announced by a node running both the attack detection and the region-based mitigation. Upon detecting an attack, this node would automatically distribute the provider records across the entire mitigation region. The requesters, which verifies if the content was eclipsed, were unmodified Kubo implementations, since the defense mechanism only considers the publication process. These nodes searched for content every 30 minutes and were restarted every 60 minutes. Content was considered unreachable if no response was retrieved within 30 seconds.

In a real-world scenario, content is published before Sybil nodes are instantiated. However, in our experiment, we chose to instantiate the malicious nodes before the publication in order to avoid waiting for the initial expiration cycle, which would delay the results by 48 hours. While the results would be similar, the effectiveness of the attack would be reduced during the first two days.

D. Evaluation

The results of our attack are illustrated in Figure 8, which presents the retrieval eclipse percentage over a 15-day period. By the end of the second day, the eclipse rate had already reached nearly 50%. This rate continued to increase over time, due to the adaptation period required for the Sybil nodes. As the routing table favors stable and long-standing connections, the malicious nodes must wait for opportunities to be incorporated into more routing tables.

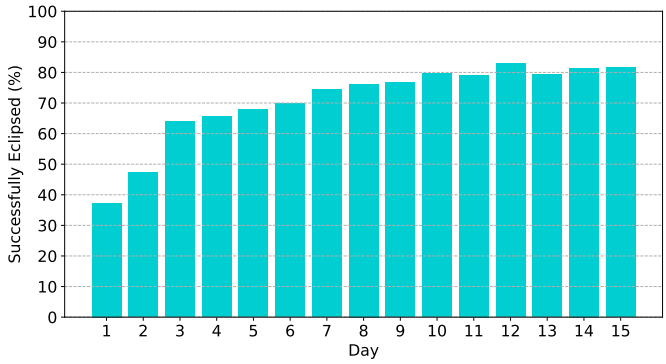

Fig. 8: Active attack using optimized Sybil placement against region-based mitigation.

Starting on the 10th day, the results began to stabilize, converging to an average eclipse rate of 80%. By the end of the experiment, this rate had increased slightly to 82%. Although the test was extended by an additional 15 days, the results remained consistent and are, for this reason, not included in the figure.

Since the attack successfully bypasses the proposed detection mechanism and continues to eclipse 80% of content requests, we consider the mitigation effective in the passive scenario, but vulnerable to our active attack. Therefore, in the following section, we introduce a more lightweight and reliable strategy for defending against Sybil attacks in DHTs: the SR-DHT-Store.

V. Sybil-Resistant DHT Store

This section introduces a new defense mechanism for mitigating Sybil attacks in DHTs: the Sybil-Resistant DHT Store (SR-DHT-Store). The proposed approach is designed to be executed during every publication, securing the provide operation while imposing minimal overhead. We begin by describing the defense mechanism, followed by an analysis of its internal structure, and conclude with an evaluation of its performance and associated costs.

A. Description

We rely on the fact that PIDs are evenly distributed across the Kademlia address space. For a network size of N nodes, we can determine the limit distance d_k at which at least k of the closest nodes should be found by simply dividing the space by the total number of nodes. However, this calculation requires to know the network size, which is a challenging value to obtain in a fully decentralized system where nodes lack a global view of the network. Consequently, estimating the network size involves many measurements on the DHT that generate overhead. Since the network size is only used to deduce the distance d_k, an alternative approach is to estimate directly the average distance to the k-th farthest node by sending simple queries to random nodes in our routing table. Specifically, we send a FIND_NODE(PID) request to a peer searching for its own PID, in order to retrieve its k closest peers. With approximately Q_{d_k} direct exchanges, we can obtain a sufficiently accurate estimate of d_k. This estimated distance can be further refined using the results of lookups naturally performed by nodes, such as those used to update their routing tables.

When performing a lookup to identify the k closest nodes during publication, and given that d_k has been previously calculated, we can opportunistically send provider records to any nodes found that are closer than the calculated distance during the lookup. This exchange can be executed in parallel while continuing the lookup process. After completing the lookup, if the total number of sent records is below the replication factor k, additional records are distributed to nodes beyond d_k until k peers store the data. In the presence of an attack, both reliable and Sybil nodes will receive the records, causing the total number of resolvers to exceed k by approximately the number of instantiated Sybil nodes. The mitigation process is illustrated in Figure 9.

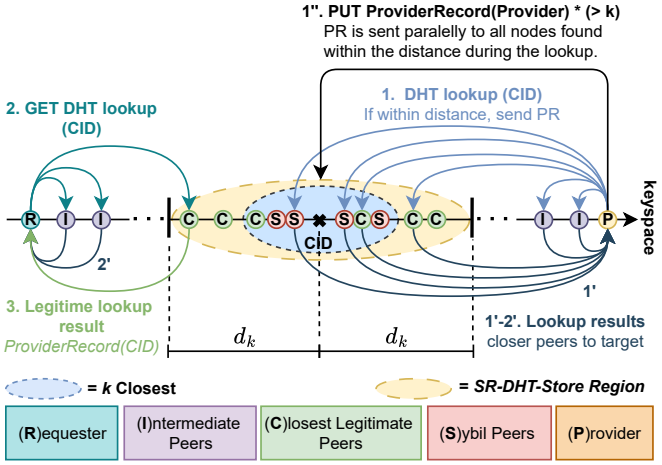

Fig. 9: SR-DHT-Store opportunistic publication. This example assumes $k = 6$.

To sum up, our approach differs from Kademlia [2] in that we no longer rely on the closest nodes to a CID, which favors Sybil Attacks. We diverge from the approach of Sridhar *et al.* [6] as we do not seek to detect the attack, since the detection can be cheated, but rather we improve the way region-based queries can be systematically performed, as explained in the next sections. Moreover, our distance estimation uses the XOR metric instead of the CPL. This provides more precision by considering all 256 bits rather than only the most significant bits. This metric is free from threshold effect, as the zone defined by the CPL is limited by the powers of two, only doubling or halving according to the estimated network size.

Another drawback of using CPLs is that an inaccurate estimation can lead to additional nodes being contacted to store records in case of attack detection or false positives. As of August 2024, based on the average network size, the estimated CPL typically falls around 9. This value is generally accurate, containing an average of $k = 20$ nodes within the defined region. However, as nodes continuously join and leave

the network, the region boundary may fall between two CPL values, leading to incorrect region delimitation and resulting in significantly more or fewer than k nodes receiving the record.

B. Zone Delimitation

While an accurate estimation of the average d_k value is crucial for our mitigation, this calculation must also be efficient to minimize overhead. We identified two methods for determining the distance to the k-th farthest node from random peers in order to calculate the average d_k: (i) directly requesting their k closest nodes, and (ii) performing a lookup toward them. Due to the structure of the routing table, which is organized into k-sized buckets, nodes store more information about their closest peers than about the rest of the network. Our approach takes advantage of this by directly querying nodes for their k closest peers. While this method is not as precise as a lookup, it provides a sufficiently accurate estimation with minimal overhead.

To evaluate this trade-off, we compared the k closest nodes obtained from both approaches. On an average of 100 tests, 16.32 among the $k = 20$ closest nodes were identical in both methods. Despite this minor difference, each lookup involved querying an average of 42.3 nodes until reaching termination, whereas a direct query required only a single request. For this reason, when joining the network, we opted to obtain the initial average d_k value by querying Q_{d_k} peers for the closest nodes they know and calculating the average distance of the k-farthest node.

In Figure 10, we evaluate the impact of the quantity Q_{d_k} on the calculation of d_k by initializing Q_{d_k} with values ranging from 1 to 20 queries toward randomly generated nodes. Each value within the range $1 \leq Q_{d_k} \leq 20$ was tested ten times, and the results presented are the averages of all measurements. To assess the accuracy of each estimation, we compared the calculated distance against a database of all nodes discovered during 100 random lookups, determining how many provider records would theoretically be sent for each Q_{d_k}-based estimation. Observing the chart, while $Q_{d_k} = 1$ and $Q_{d_k} = 2$ produced results close to k due to chance, starting from $Q_{d_k} = 10$, the estimations began to present minor variations. For this reason, upon initialization, a node must first query $Q_{d_k} = 10$ peers to compute an initial average d_k.

The average distance to the k-farthest peer must be updated over time to improve the accuracy and take into account the variation of the network size in time. Typically, P2P networks are subject to hourly size variation which magnitude depends on the proportion of personal computers hosting peers using public IP addresses. To maintain an accurate average d_k, each time a lookup is performed, the distance of the k-farthest node from the response should be incorporated into the d_k calculation. As explained in II-C, the routing table is updated every ten minutes through $L_{d_k} = 16$ lookups, which can be used to refine our average distance estimation with no additional overhead. To integrate new measurements while preserving the influence of previous values, we propose using an Exponentially Weighted Moving Average (EWMA). This approach allows updates while

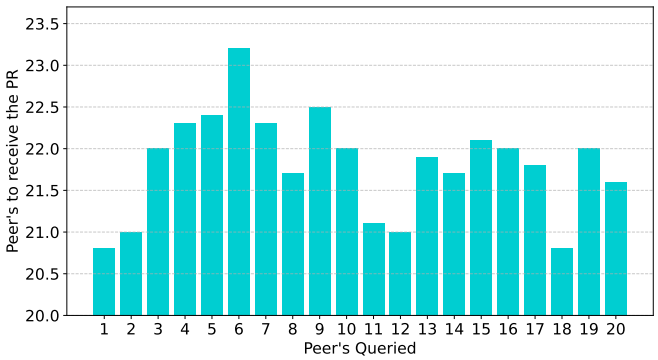

Fig. 10: Impact of Q_{d_k} on the number of peers receiving provider records.

maintaining stability in the estimation. The updated distance can be calculated using the following equation:

$$S_{t+1} = y_t \cdot \alpha_{sf} + S_t(1 - \alpha_{sf}). \tag{4}$$

The variable y_t represents the most recent measurement, while S_t denotes the last calculated moving average. The smoothing factor α_{sf}, where $0 \leq \alpha_{sf} \leq 1$, determines the influence of previous averages on the current measurement. According to Hunter [31], the optimal value of α_{sf} should be selected by minimizing the mean squared error (MSE) relative to the target value, in our case, sending exactly k provider records. Hunter also recommends computing the initial average as a simple mean over 4–5 values, which corresponds with our approach, where we initially estimate d_k using $Q_{d_k} = 10$ queries. To determine the optimal α_{sf} for our scenario, we first estimated d_k using $Q_{d_k} = 10$, then observed the MSE resulting from refining this estimate with $L_{d_k} = 16$ additional lookups across a range of α_{sf} values, from 0.001 to 0.999 in increments of 0.001. The lowest average mean squared error was observed at $\alpha_{sf} = 0.100$.

In the rest of the paper, we estimate d_k by measuring the distance to the k-th farthest node from the results of the k closest nodes from $Q_{d_k} = 10$ queries and $L_{d_k} = 16$ additional lookups. The moving average is updated with a weighting factor of $\alpha_{sf} = 0.1$. Using this approach, we consider both the initial estimation and the subsequent lookups performed by the node to initialize its RT, similarly to a real-world scenario.

C. Cost Analysis

To analyze the cost of our solution, we evaluated the accuracy of d_k estimation using our database of 100 random DHT lookups to determine how many nodes would have received the provider records according to the distance d_k we estimate. The distance calculation and verification were performed ten times, and the results are presented in Figure 11, divided into two charts: one showing the initial distance estimation and the other presenting the refined estimation with EWMA. On the x-axis, we display the average distance calculated across the ten tests for both the initial and refining phases. All data from each of

the ten tests were included in the graph, meaning that each box plot represents 10 (tests) $\times$ 100 (lookups per test) $=$ 1000 data points. Each chart also includes the equivalent CPL-based approach, which corresponds to sending provider records to all nodes within the $keyspace - \lfloor \log_2(d_k) \rfloor$ CPL.

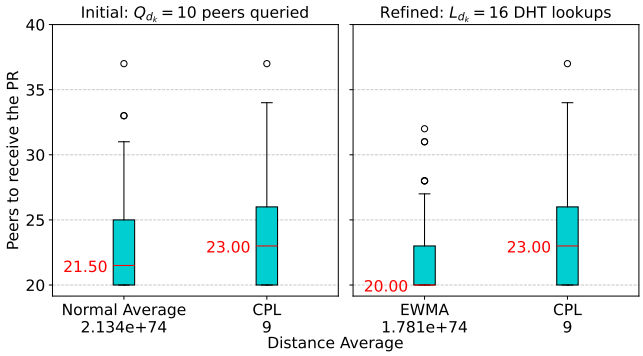

Fig. 11: Average number of nodes receiving PRs when initializing d_k with Q_{d_k} queries and refining with L_{d_k} lookups.

By querying only Q_{d_k} nodes for their k-th farthest peer, we observed a median of $k+1.5$ nodes receiving the provider record. After refinement, the median matched exactly k. In a real-world scenario, any distribution containing fewer than k nodes within the distance d_k would pursue the publication process of provider records after the lookup phase until the target of k is reached by considering slightly farther peers. As an additional cost, we consider only the extra provider record exchanges that exceed k, as k records are already sent during a standard provide operation.

Before and after refinement, the average number of extra nodes contacted was $C_{XOR_i} = 2.786$ and $C_{XOR_r} = 2.064$, respectively. If we instead used the CPL of the nodes to define the target zone, the number of contacted nodes would increase to $C_{CPL_i} = 5.135$ and $C_{CPL_r} = 3.860$. For a newly instantiated node, the average extra cost of executing our mitigation is given by $C = Q_{d_k} + C_{XOR_i}$, resulting in an overhead of 12.786 contacted nodes. For a node already established in the network, regular lookups naturally refine the precision of d_k, reducing the extra cost to $C_{XOR_r} = 2.064$ nodes per publication.

Comparison against Sridhar *et al.* [6]. The amount of nodes contacted at each stage of both mitigation strategies is compared in Table I. The table highlights individual phases, and shared phases, such as the main lookup performed to find the k closest nodes to the content.

Immediately after node initialization, their attack detection mechanism requires estimating the network size. According to their implementation on Kubo, available in their paper, this process involves performing 10 lookups to gather network data, and following our experiments, this resulted in contacting an average of 433.8 nodes. We avoid this overhead by directly calculating the distance to the k-th closest peer. By using this approach, we eliminate the need to estimate the network size in order to derive a value that is already available from the lookup responses, the d_k.

TABLE I: Number of nodes contacted during each phase of Region-Based Queries and SR-DHT-Store.

Phase		Attack Detection + Region-Based Queries	SR-DHT-Store
Network Estimation after Bootstrap		433.8	N/A
d_k Estimation		N/A	10
Lookup		42.3	
Region-Based Queries		72.9	N/A
False Positives		$72.9 \times 0.11 \approx 8$	N/A
PR Overshoot		N/A	2.7 to 2
New Node	No Attack	484.1	**55**
	Attack	549	**55**
Established Node	No Attack	50.3	**44.3**
	Attack	115.2	**44.3**

The full list of peers discovered during the main lookup, performed by any node to locate the k closest nodes to the content during publication, are reused in our case to eliminate the need for a specific region crawl. In a regular Kademlia scenario, only the k closest nodes from the query results are used, however, it is possible to retain all discovered nodes during the lookup, avoiding the need to search for them again. To compare the effectiveness of region-based queries with the peer list obtained from a standard lookup, we applied both approaches to identify all nodes within at least CPL 9 of ten randomly chosen identifiers. On average, the region-based queries found 22.9 nodes, while the standard lookup returned 22.7 nodes. For this reason, we consider the entire cost of the region-based approach as additional because unnecessary, since the results from the standard lookup are obtained at no extra cost, as this operation is already required for a standard content publication.

In conclusion, when accounting for the false positive rate of the attack detection that we evaluate at 11%, as shown in Figure 5, and the initial PR overshoot of SR-DHT-Store, starting at 2.7 nodes and reduced to 2 nodes after refinement, our mitigation strategy reduces the number of contacted nodes by approximately 88.6% in the absence of an attack, and by nearly 90% during an attack. When the node is already established in the network, the number of contacts is reduced by 10% under normal conditions and by 63% in the presence of an attack.

D. Evaluation

The SR-DHT-Store was evaluated against both the passive k closest attack and the active optimized Sybil placement attack. All Sybil nodes shared the same implementation, with configurable flags to define their behavior, enabling them to switch between passive and active modes. In this subsection, we present the results of the tests conducted to assess the effectiveness of our mitigation, which are illustrated on Table II.

TABLE II: Evaluation of each mitigation strategy against the passive k closest and the active optimized Sybil positioning attacks.

Attack	Retrieval Success Rate (%)	
	Attack Detection + Region-Based Queries	SR-DHT-Store
Passive k Closest	100%	100%
Active Optimized Sybil Positioning	18%	28%

We first performed a passive k closest attack on the latest Kubo release, positioning Sybil nodes using the same strategy as in previous IPFS attacks [6], [13]. We targeted five distinct CIDs, performing five lookups toward each of them after publishing using both the SR-DHT-Store and the attack detection + region-based queries. The file was successfully retrieved in every lookup, resulting in a 100% success rate using both mitigations.

For the optimized Sybil placement attack, we reused the same Sybil nodes deployed in our earlier active attack experiment, in Section IV-D, to evaluate our mitigation against well-established adversarial nodes. Both mitigations were tested against the same group of Sybils, so for each CID and each mitigation, the content was republished through different peers. During content retrieval, if the specific mitigation provider was not included among the providers returned in the records, we considered the content to be eclipsed, as the response originated from a Sybil node. Similarly to the passive attack scenario, each test was repeated five times using each mitigation, for a total of ten tests. As result, only 28% of the lookups successfully retrieved the content after providing with our SR-DHT-Store. The region-based queries performed a standard lookup mechanism, as they initially failed to detect the attack, retrieving the content in only 18% of cases.

The rationale is the following. Since some active Sybil nodes are among the closest to the content, they can still be the first contacted during content retrieval, leading early search abortion even when more other nodes in the network hold the true records. Our mitigation focuses on securing the store procedure to ensure that provider records are not exclusively received by malicious nodes. However, the lookup process remains vulnerable to active attacks and requires further enhancements to address this threat. In the following section, we introduce additional mitigation strategies and defense mechanisms that can complement our approach, along with their corresponding evaluation.

VI. Enhanced Mitigation Strategy

While the content publication process is secured by SR-DHT-Store, the lookup procedure remains vulnerable to adversaries exploiting the early termination. To address this, we propose to combine in this section SR-DHT-Store with two client-side defense mechanisms. Both lookup defenses were implemented in Kubo and evaluated against our active attack model. An additional mechanism is also proposed in this section, however, it was not implemented, as we believe it increases the difficulty of performing the attack but does not fully mitigate the issue.

PR Limitation. No external node should have the ability to prematurely terminate a DHT lookup. Currently, $PR_{\max} = PR_{\max_{peer}} = 10$, meaning a peer can respond to a `FIND_VALUE` request with up to ten provider records, just enough to externally halt the lookup process. To prevent this behavior, setting $PR_{\max_{peer}} > PR_{\max}$ ensures that the node continues searching until it has locally collected $PR_{\max}$ provider records. This approach preserves compatibility between peers by leaving $PR_{\max_{peer}}$ unchanged and only modifying $PR_{\max}$ locally to determine when to terminate the lookup based on the accumulated records.

In our implementation, we tested two different values for $PR_{\max}$: 50 and 200. These values correspond to requiring complete sets of ten provider records from 5 and 20 peers, respectively, in order to terminate the lookup.

Disjoint Lookup Paths. This secure lookup mechanism, proposed by S/Kademlia [14], was introduced in Section III-C. In the standard Kademlia k closest search, α nodes are queried in parallel, but all discovered peers are added to a shared query list. This design allows a single adversarial node, once queried, to potentially compromise the entire search. To mitigate this risk, we used $d = 3$ disjoint lookups, where the results of each request are kept entirely separate. These disjoint lookups must also ensure that no node is contacted more than once across the different paths. Although this approach introduces additional overhead, it is only necessary when searching for content, rather than for every lookup operation. To further reduce overhead, the disjoint lookups could be used only in cases where the content was not retrieved in the first attempt.

IP Address Limitation in Lookup. During a DHT lookup, no more than one node sharing a same IP address, or even a given IP sub-network prefix size, should be contacted. This limitation increases the difficulty for attackers, as it requires a larger number of unique IP addresses to successfully carry out an attack. When combined with disjoint lookup paths and the PR limitation, it adds another layer of complexity, forcing the attacker to distribute the Sybils across multiple networks. While we believe this measure increases the effort required for launching an attack, it is not a definitive solution as the other proposed mitigations. For this reason, it was not implemented or evaluated in our experiments.

A. Cost Analysis

The PR Limitation introduces no additional cost compared to a normal lookup when no attack is present. Since all k closest nodes will typically return valid provider records for the content, the requesting peer will not reach $PR_{\max}$ and will follow the standard termination procedure. In the case of an active attack, each contacted Sybil node can result in ten additional lookups for the fake PIDs. This occurs because the node is unable to contact the providers returned by the malicious peers and must perform a separate lookup for each false provider in an attempt to locate a valid, contactable IP address.

The other mitigation, disjoint requests, introduces an overhead approximately 2.5 times the cost of a normal lookup, as it performs $d = 3$ disjoint lookups. To measure this cost, we conducted ten lookups using using the standard strategy and the disjoint request approach, targeting a provided content. When searching for a previously provided content, the standard lookup queried an average of 21.9 nodes before retrieving a valid provider record, while the disjoint requests contacted an average of 55.7 nodes.

B. Evaluation

All tests were conducted using the same Sybil nodes previously used in the evaluation of the optimized Sybil placement attack, in Section IV-D. We began by testing only the mitigations applied during the content publication phase, followed by a standard lookup for retrieval. Next, we published the content using SR-DHT-Store and attempted retrieval by applying both the PR limitation, with thresholds of 50 and 200, and disjoint lookup paths. Each retrieval attempt was repeated five times for each of the five targeted content items. Our results are illustrated in Figure 12.

The results presented in the first two columns of the figure are identical to those shown in Section V-D. The last three columns combined the lookup defense mechanisms alongside the SR-DHT-Store mitigation. When protecting the lookup using a PR limitation with a threshold of 50, the content was successfully retrieved in 88% of cases. This demonstrates improved resistance, but also highlights the persistence of Sybil nodes, as in 12% of the lookups five Sybils were among the first nodes queried. By increasing the PR threshold to 200 or using disjoint lookup paths, the content was retrieved in 100% of the cases, fully mitigating our attack strategy.

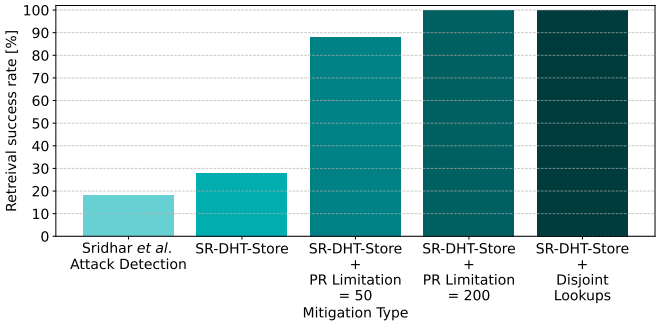

Fig. 12: Effectiveness of mitigations against the optimized Sybil positioning attack.

C. Discussion

When combined with the SR-DHT-Store, the PR Limitation works well against active attack strategies, while disjoint lookups can be applied against both active and passive attacks, but also against adversarial routing [24]. We consider the PR Limitation a simple and efficient solution that could be implemented to counter active attacks, at almost no cost. The disjoint lookups offer a more robust and general solution that brings diversity in the result of a DHT search to better withstand Sybil attacks. While disjoint requests increase the cost of performing lookups, they are capable of fully mitigating Sybil attacks on DHTs when applied after the SR-DHT-Store. In conclusion, we highlighted synergies between our provide mitigation and two other lookup defense mechanisms that leave some choice for IPFS developers to select the defense strategy according to the overhead they tolerate.

VII. Conclusion

In this paper, we presented a new active Sybil attack targeting Kubo, the largest IPFS implementation. The attack was evaluated against the latest mitigation proposed by Sridhar *et al.* [6], which, although well implemented in a forked version, has not yet been merged into the official repository. Our attack achieved a success rate of 82% against their mitigation by exploiting an early termination in the libp2p lookup process and an optimized placement of Sybils to stay below the detection threshold triggering the mitigation. When executed against the current official version of Kubo, the attack is capable of completely eclipsing content, similar to the previously identified passive Sybil attack on the network.

To mitigate the attack, we introduced the SR-DHT-Store, a low-cost defense mechanism executed during each content publication. While our approach effectively addresses previously identified passive attacks on the network at a lower overhead, it does not, on its own, fully mitigate the active attack presented in this work. To achieve complete protection, we introduced two additional defense mechanisms designed to secure the content requester. When combined with our new publication strategy, both mechanisms fully mitigated the active attack.

In future work, we plan to explore new active attacker models that aim to disrupt the network not regarding the DHT's behavior, but rather hijacking what is stored in it. Our goal is to counter content pollution by designing new consensus mechanisms that rely on the responses diversity enabled by SR-DHT-Store. Another work would be to refine the zone considered by our publication mitigation by introducing a minimum distance threshold, $d_{k_{\min}}$, relative to a CID. Any peer closer than this threshold would be considered statistically too close and excluded from lookup results.

VIII. Ethics Consideration

While this paper introduces a new active attack approach targeting IPFS, the recently discovered passive attack remains unmitigated in the current Kubo implementation. Therefore, this experiment does not introduce any new threats beyond those already present in the system. Since the mitigation proposed by Sridhar *et al.* [6] has not yet been integrated into the mainstream client, we do not consider this to be an attack on Kubo itself.

All targeted content was provided exclusively by nodes under our control. No legitimate content in the network was disrupted or eclipsed, since our malicious nodes responded normally to all requests except those specifically targeting the test content. To prevent instability in the DHT, the requester nodes operated

in client mode, minimizing the impact of their joining and leaving the network.

References

[1] J. R. Douceur, "The Sybil Attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. Springer-Verlag, 2002, pp. 251–260.

[2] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. Springer-Verlag, 2003, pp. 53–65.

[3] M. Steiner, T. En-Najjary, and E. W. Biersack, "Exploiting KAD: possible uses and misuses," vol. 37, no. 5, pp. 65–70, 2007. [Online]. Available: https://dl.acm.org/doi/10.1145/1290168.1290176

[4] T. Cholez, I. Chrisment, and O. Festor, "Monitoring and Controlling Content Access in KAD," in *2010 IEEE International Conference on Communications*, 2010, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/5502179

[5] T. Cholez, I. Chrisment, O. Festor, and G. Doyen, "Detection and mitigation of localized attacks in a widely deployed P2P network," vol. 6, no. 2, pp. 155–174, 2013. [Online]. Available: https://doi.org/10.1007/s12083-012-0137-7

[6] S. Sridhar, O. Ascigil, N. Keizer, F. Genon, S. Pierre, Y. Psaras, E. Rivière, and M. Król. (2023) Content Censorship in the InterPlanetary File System. [Online]. Available: http://arxiv.org/abs/2307.12212

[7] J. Benet. (2014) IPFS - Content Addressed, Versioned, P2P File System. [Online]. Available: http://arxiv.org/abs/1407.3561

[8] "ipfs/kubo," IPFS Project, 2025. [Online]. Available: https://github.com/ipfs/kubo

[9] Berty · Berty Technologies. Berty Technologies. [Online]. Available: https://berty.tech

[10] DTube. [Online]. Available: https://d.tube/

[11] Filecoin. A Decentralized Storage Network for the World's Information. Filecoin. [Online]. Available: https://filecoin.io/

[12] libp2p - A modular network stack. libp2p. [Online]. Available: https://libp2p.io/

[13] T. Cholez and C.-L. Ignat, "Sybil Attack Strikes Again: Denying Content Access in IPFS with a Single Computer." ACM, 2024, p. 1. [Online]. Available: https://inria.hal.science/hal-04666290

[14] I. Baumgart and S. Mies, "S/Kademlia: A practicable approach towards secure key-based routing," in *2007 International Conference on Parallel and Distributed Systems*, 2007, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/4447808

[15] Merkle Directed Acyclic Graphs (DAG) — IPFS Docs. [Online]. Available: https://docs.ipfs.tech/concepts/merkle-dag/

[16] Multiformats. [Online]. Available: https://multiformats.io/

[17] Work with pinning services — IPFS Docs. [Online]. Available: https://docs.ipfs.tech/how-to/work-with-pinning-services/

[18] Multiaddr — Multiformats. [Online]. Available: https://multiformats.io/multiaddr

[19] go-libp2p-kad-dht/amino/defaults.go at master · libp2p/go-libp2p-kad-dht. [Online]. Available: https://github.com/libp2p/go-libp2p-kad-dht/blob/31c361257e16379b6dee2fc2981f75ea0935a102/amino/defaults.go

[20] Distributed Hash Tables (DHT) — IPFS Docs. [Online]. Available: https://docs.ipfs.tech/concepts/dht/

[21] Bitswap — IPFS Docs. [Online]. Available: https://docs.ipfs.tech/concepts/bitswap/

[22] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse Attacks on Overlay Networks: Threats and Defenses," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/document/4146884

[23] P. Wang, J. Tyra, E. Chan-Tin, T. Malchow, D. F. Kune, N. Hopper, and Y. Kim, "Attacking the Kad network," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, ser. SecureComm '08. Association for Computing Machinery, 2008, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/1460877.1460907

[24] M. Kohnen, M. Leske, and E. P. Rathgeb, "Conducting and Optimizing Eclipse Attacks in the Kad Peer-to-Peer Network," in *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, ser. NETWORKING '09. Springer-Verlag, 2009, pp. 104–116. [Online]. Available: https://doi.org/10.1007/978-3-642-01399-7_9

[25] B. Prünster, A. Marsalek, and T. Zefferer, "Total Eclipse of the Heart – Disrupting the InterPlanetary File System," 2022, p. 3735. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/prunster

[26] (2020) Hardening the IPFS public DHT against eclipse attacks. IPFS Blog & News. [Online]. Available: https://blog.ipfs.tech/2020-10-30-dht-hardening/

[27] Peer Diversity for Routing Table and Querying by aarshkshah1992 · Pull Request #88 · libp2p/go-libp2p-kbucket. GitHub. [Online]. Available: https://github.com/libp2p/go-libp2p-kbucket/pull/88

[28] Release v0.34.0 · libp2p/go-libp2p. GitHub. [Online]. Available: https://github.com/libp2p/go-libp2p/releases/tag/v0.34.0

[29] D. Pisinger, "Where are the hard knapsack problems?" vol. 32, no. 9, pp. 2271–2284, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030505480400036X

[30] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8032

[31] J. S. Hunter, "The Exponentially Weighted Moving Average," vol. 18, no. 4, pp. 203–210, 1986. [Online]. Available: https://doi.org/10.1080/00224065.1986.11979014