An Empirical Study of Storj DCS: Ecosystem, Performance, and Security

Hao Li[1], Xianghang Mi[3*], Yanzhi Dou[4], Shanqing Guo[12*]
[1]Shandong University, Qingdao, China. [2]Quancheng Laboratory, Jinan, China.
[3]University of Science and Technology of China, Hefei, China. [4]Independent Researcher.
202020883@mail.sdu.edu.cn, xmi@ustc.edu.cn, aaron.yzdou@gmail.com, guoshanqing@sdu.edu.cn

***Abstract*—In the age of pervasive computing, traditionally centralized cloud storage (CCS) services may not fit in well due to their centralized architecture, limited worldwide availability, high expense, and security and privacy concerns. To address these issues, decentralized cloud storage (DCS) services emerged recently. However, previous works focus on analyzing the technical design of DCS services, e.g., their incentive mechanisms. Little is known regarding how well these DCS services work in real-world operations. In this paper, we fill this gap by providing the first empirical measurement of Storj, one of the most extensive in-operation decentralized cloud storage services, focusing on its ecosystem, performance, and security implications.**

Our study is made possible through multiple measurement techniques to automatically capture storage nodes, profile Storj's quality of service, understand co-located network threats, and evaluate potential attacks in a simulated environment. Leveraging these techniques, a set of insightful findings have been distilled. Particularly, we have observed over 32K storage nodes as well as 155K unique node IP addresses, which are widely distributed in 122 countries, 2,418 ASNs, and 205 /8 IPv4 prefixes. Regarding performance, storage customers located in Europe or the United States tend to enjoy a better storage performance than those in Asia-Pacific, likely due to the imbalanced distribution of storage nodes in different regions. Lastly, what is concerning is that 4.48% of IPs of storage nodes were found to have been associated with various malicious activities, especially botnets and cryptomining. Another vulnerability is that a malicious storage node could exploit multiple channels to boost its storage reputation while demoting that of benign nodes.

***Index Terms*—Decentralized cloud storage, Storj, measurement, performance, security**

I. Introduction

Cloud storage has been increasingly adopted during the last decade by both individuals and businesses, e.g., traditional cloud storage services such as Google Cloud Storage (GCS) [1], and Amazon Simple Storage Service (AWS S3) [2]. These centralized cloud storage (CCS) services are widely used in many scenarios, such as file hosting, data backup for devices, and storing sensitive business data. However, CCS services suffer from a common set of notable limitations including the outage issue [3], security and privacy concerns (e.g., censorship on the stored data), data silos, etc [4].

In recent years, decentralized cloud storage (DCS) services (such as Storj [5], Sia [6], and IPFS Filecoin [7]) emerge to address these limitations. Across DCS networks, storage nodes are responsible for data storage and retrieval. Different from CCS services wherein all storage nodes are operated by the service provider, storage nodes in DCS networks are operated by different parties and tend to a global distribution. Given the rise of DCS services, several works [4], [8]–[10] have been conducted in the last few years, jointly providing a theoretical overview of DCS services. However, little is known regarding how well DCS services work in the real world, in terms of ecosystem, performance, and security. To fill in this gap, we carry out this empirical and extensive measurement study on Storj, one of the most representative DCS services. Among DCS services, Storj has the largest number of storage nodes as well as one of the largest usage volumes (as depicted in §II). Also, it is the only DCS service that supports storage access through S3 APIs [11], which allows us to carry out a direct comparison between Storj and CCS services.

In our study, we aim to answer the following research questions. First of all, we want to profile storage nodes, the most critical component that distinguishes a DCS service from its centralized counterpart. We will gain a better understanding of how Storj's storage nodes are distributed around the world and how they change and develop over time. Then, we want to empirically investigate the quality of service (QoS) that the Storj network can achieve, whether its QoS is comparable with that of CCS services (e.g., AWS S3). Similarly, since the fact that security and privacy are major distinguishing features of Storj, it is worthwhile to profile the security implications of Storj in a way that is not only qualitative but also quantitative.

Answering these research questions is non-trivial and several technical challenges should be addressed. First of all, no metadata datasets are available for us to learn details of daily active storage nodes. We thus design and build up a storage node collector which can automatically interact with Storj's satellite (one of the primary components of the Storj network) and query about the latest list of all active available storage nodes. Besides, to extensively profile the QoS performance of the Storj, many factors should be considered, e.g., client-side bandwidth, the locations of both the clients and the Storj satellites, various file operations (upload/download), and different file sizes. We thus build up a performance evaluation framework that can automatically carry out and collect logs for control experiments following given configurations. Furthermore, to investigate the fundamental security risks that are inherent in Storj's design, we also build upon the Storj testbed

*Corresponding authors.

979-8-3503-9973-8/23/$31.00 ©2023 IEEE

so as to simulate the Storj network in a realistic manner and evaluate the feasibility of a series of vulnerability scenarios.

Leveraging the methodologies mentioned above, we carry out in this study, an empirical characterization of the Storj network for the first time, along with a set of novel and insightful findings distilled. First of all, during our 16-month measurement between April 2021 and August 2022, we observed 32,881 unique Storj storage nodes which are hosted on 155,457 unique IP addresses residing in 122 countries, 2,418 ASNs, and 205 /8 IPv4 prefixes. And also, we have looked into the lifetime and churn rates of storage nodes with a 9.6% month-by-month churn rate on average.

Besides, we have profiled the quality of service of the Storj network. It turns out that customers located in Europe or the United States can enjoy a better storage performance compared to ones in Asia-Pacific, which is likely due to the imbalanced distribution of storage nodes in different regions. When compared with traditional centralized cloud storage systems (GCS [1] and AWS S3 [2]), Storj was comparable in quality of service measured by latency and throughput of download in Europe but still fell behind in the operation of upload. However, in terms of pricing, Storj has great advantages over the two centralized cloud storage services.

Lastly, we investigated the security implications of the Storj network with a focus on its storage nodes since these nodes are operated by untrusted third parties. Among all the IPs of storage nodes, 4.48% were associated with various malicious activities, e.g., botnets and suspicious cryptomining. This can compromise the reputation and availability of the involved storage nodes, leading to the blocking of storage traffic, and ultimately undermining the reliability of the Storj network. We have also identified multiple channels for a dishonest or even malicious storage node to manipulate its storage reputation, in an attempt to impede Storj's reputation system which can increase its chance of getting selected to serve more data storage orders. We refer to such scenarios as the reputation manipulation vulnerabilities, for which we have demonstrated their feasibility in a simulated testbed.

Our main contributions can be summarized as follows.

•*New techniques*. We have first designed and implemented several new techniques to empirically profile the Storj network from various aspects. In particular, a storage node collector is built up for automatically capturing active storage nodes. Then, we presented an evaluation framework for comprehensively measuring the performance as well as comparing Storj with CCS services. Lastly, we have explored a reputation manipulation vulnerability to investigate the fundamental protocol security risk that is inherent in Storj's design.

•*New findings*. We are the first to study Storj DCS and a set of insightful findings have been distilled in our study regarding landscape and evolution, performance, and security implications.

II. Background of Storj

Before stepping into our measurement methodology and respective results, we start by presenting necessary background knowledge, especially the fundamental concepts of various decentralized cloud storage networks and their pros and cons, as well as the unique characteristics of Storj.

TABLE I
AN OVERVIEW OF DIFFERENT DCS NETWORKS.

System	Storage Capacity[1]	Active Storage Nodes	Durability[2]	S3
Storj [5]	18.8PiB	16K	Erasure Codes	✓
IPFS Filecoin [7]	18EiB	4K	Replication	✗
Sia [6]	7.43PiB	0.5K	Erasure Codes	✗

[1] Storage Capacity: Storage network stored data scales queried in March 2022.
[2] Durability: Methods to make stored files persist.

Decentralized Cloud Storage. To address the aforementioned constraints of CCS services, decentralized cloud storage services emerged in recent years. Typical services of DCS networks include IPFS Filecoin [7], Sia [6], and Storj [5].

Across these DCS networks, storage nodes play a crucial role in data storage and retrieval. Different from CCS services wherein storage servers are operated and controlled by the same owner (the service provider), storage nodes in DCS networks source from different parties and have a global distribution. Across storage networks, storage nodes will be paid for storing data and serving data retrieval, and the payment is calculated upon multiple factors including the size of the data to be stored, the storage period, and bandwidth incurred by data uploading and downloading, etc. To audit the operations of storage nodes, various proof-of-work schemes have been adopted, e.g., proof of replication, and proof of retrievability [12]. Also, to execute and record storage or retrieval transactions, Filecoin and Sia leverage smart contracts while Storj utilizes centralized servers called *Satellites*.

Why Storj. In particular, Storj and Sia have built-in support for data redundancy leveraging various Erasure Coding schemes [13]. Briefly, a file will be divided into many shards before being uploaded to distinct storage nodes, and only a subset of the resulting shards is necessary to recover the original file. In addition, shards in Storj and Sia are encrypted before uploading, and the encryption keys are only known by the file owner. Different from Storj and Sia, by the time of our paper writing, Filecoin has yet to support erasure coding and client-side encryption. By default, a file is stored by a single storage provider in Filecoin without client-side encryption [14]. Furthermore, Storj also has the largest number of storage nodes and the second-highest storage capacity. And different from the other DCS services, Storj provides S3-compatible gateways for clients to access their files through S3 APIs, which makes it easier for users of centralized storage services to migrate to Storj, as well as makes it possible to compare Storj's performance directly with CCS services (such as AWS S3). Table I presents a comparison of these DCS services, and we thus choose Storj as our research subject, considering these factors.

The Technical Design of Storj. We provide a technical overview of Storj as follows.

Participants. The Storj storage network consists of three key

components. One is the *Storage Clients*, the client-side programs (e.g., Storj Uplink CLI[1] and S3 CLI [11]) leveraged by storage customers to access this storage network and carry out storage operations such as uploading and downloading. Then there are the *Storage Nodes* which are responsible for storing data and earning payments. A storage node can be operated by any third party, trustworthy or not. The third is *Satellites* which are dedicated to many tasks including storage node discovery, data audit and repair, and billing, among others. For instance, the auditing system periodically verifies that storage nodes actually store the data they agreed to store and evaluates the node's reputation score (which is very closely related to the rewards). Currently, all satellites are operated in a centralized manner by the official Storj administrator.

File Uploading and Downloading in Storj. When an Uplink client uploads a file to the Storj network, the file will be first split into segments of a predefined size (the current size is 64MB[2]) and each segment will be further encrypted locally on the client side. Given the encrypted segment, the Reed-Solomon Erasure Coding scheme [13] is applied to splitting the segment into $n = 80$ pieces. Subsequently, in order to distribute these pieces to different storage nodes and address the issue of slow nodes in the upload list, a necessary number of storage nodes ($o = 110$ until our paper writing) will be extracted from the satellites. When downloading a file stored in the Storj network, a reverse order of steps is needed. And only $k = 29$ pieces are required to recover the respective segments that were attempted to download from $l = 39$ (attempt to download pieces) different storage nodes. Specifically, the erasure coding variables are $(k, l, n, o) = (29, 39, 80, 110)$.

III. The Storj Storage Network Ecosystem

In this section, we present an overview of the Storj storage network. To start, we first present in §III-A, the storage nodes collector which is designed and deployed to automatically interact with satellites to collect active Storj storage nodes. Leveraging this storage node collector, we carried out an empirical measurement and a set of novel understandings have been distilled on Storj's landscape as introduced in §III-B.

A. The Storage Node Collector

Obtaining detailed and comprehensive metadata of storage nodes is critical for profiling the entire Storj ecosystem. Such kinds of metadata attributes for each storage node include the *node ID*, the *IP addresses*, and the *Ports* it listens to. However, no publicly available datasets contain such detailed information. And how to acquire all of the storage nodes' data without interfering with its normal operation is a very challenging thing. To fulfill this gap, we designed and implemented a storage node collector to capture currently active storage nodes in real-time. As aforementioned in §II, when uploading a file, the Uplink client will retrieve a set of active storage nodes from the satellite, so as to store the erasure-coded pieces in distinct storage nodes. We observe that the response returned by the satellite contains detailed metadata attributes for each of the storage nodes with no encryption. Also, separate requests for extracting storage nodes tend to have a different set of storage nodes returned by the satellites, which is reasonable since the satellites are designed to balance storage workloads across storage nodes.

Leveraging these observations, we built up the storage node collector by modifying the Storj Uplink CLI (v1.4.5[3]) with the necessary hooking and logging functionalities. The proposed node collector can periodically act as uploading a file to query the designated satellite server for a list of storage nodes. Then it dumps the resulting storage nodes into a local file designated through a command line parameter, before moving to the next round of storage node collection. The collector won't interact with the storage nodes in data collection. We ran our nodes collector instance during the measurement period between April 30, 2021, and August 30, 2022. The collector was configured to query the respective satellite at most once per second, which is much lower than the official rate limit of 100 requests per second for a free-tier Uplink client [15]. Also, to further reduce the request pressure of the satellites, our daily collection job will be terminated if no new storage nodes are discovered for 10 consecutive rounds. And we observed that the storage node collector runs for no more than 18 hours per day in most cases. Therefore, we believe the traffic pressure on the satellites incurred by our collector should be negligible.

Data Summary. During the 16-month measurement, our storage nodes collector captured 32,881 unique storage nodes along with 155,457 IP addresses that had ever hosted storage nodes. Although our collector was configured with a low request frequency, it turns out to be enough to capture the majority or even all of the daily active storage nodes. The anonymized dataset is released at our Github[4].

B. Landscape of Nodes

As introduced in §III-A, we deployed a collector to proactively capture storage nodes of Storj on a daily basis. From 04/30/2021 to 08/30/2022, we observed an increase in storage node IDs by 183%. Fig. 1 presents how storage nodes as well as node IPs evolve across time in terms of the volume during our collection period. We can observe a steady growth of cumulative storage nodes and their IP addresses. However, the daily active storage nodes and their IP addresses are relatively stable. Also, an interesting observation is that the number of daily active

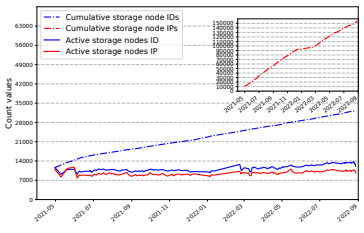


Fig. 1. The evolution of storage nodes and node IPs across time.

[1]https://github.com/storj/uplink

[2]https://docs.storj.io/dcs/concepts/definitions

[3]https://github.com/storj/uplink/releases/tag/v1.4.5

[4]https://github.com/lihaoSDU/IWQoS2023

storage nodes is constantly more than that of their respective IP addresses, which is due to the fact that a single IP address may host multiple storage nodes. One abnormal pattern revealed in Fig. 1 is that the daily captured storage nodes and IP addresses had abruptly dropped to an unexpectedly low volume for several dates, e.g., 03/11/2022 to 03/14/2022. We looked into the root cause from the Storj community and found out that the abrupt downward trend likely resulted from The Russo-Ukrainian War affecting the availability of storage nodes in both countries[5].

The Many-to-Many Relationship between Node IDs and Node IPs. We also observe a many-to-many relation between node IDs and node IPs. In other words, a storage node may migrate across IP addresses or even network blocks while a node IP may be used to host multiple storage nodes. Specifically, during our node collection period, 9,670 out of 32,881 nodes have migrated across at least 2 IP addresses, and 5,290 have migrated across 5 or more. On the contrary, among 155,457 node IPs, 13,066 have hosted more than 2 nodes while 2,218 have hosted even 5 or more nodes.

Regarding one node attaching to many IPs, many nodes were found to have attached to dynamic public IP addresses, and node operators thus adopted various DDNS services to automatically update the DNS records for their node domain names with new IP addresses. For instance, storage node "1zFzFD****ekcr" deployed Synology NAS DDNS service (*synology.me*) and 73 IP addresses were observed to address this node during our node collection period. In terms of one IP to many nodes, most of these IPs were found to be IPs of reverse port forwarding services (RPFS, e.g., *Ngrok*[6]), which are designed to expose the network services to the public without the necessity of listening to a public TCP/UDP port. The adoption of reverse port forwarding services resulted in the sharing of a small set of RPFS IPs across many storage nodes. For instance, "3.*.*.225", an IP of a Ngrok gateway server, is used to host 21 distinct storage nodes. We will make a further investigation on PFS security in §V.

Nodes Distribution. To measure the node distribution in the geolocation space and the network space, we queried IP-info [16] to extract the IP WHOIS information for the captured storage nodes. IPinfo is an IP address data provider (420 billion requests a year) to produce geolocation, ASN, hosted domains, IP range details, etc. From the IPinfo intelligence, the 155K storage nodes IPs have been distributed in 74K /24, 11K /16, and 205 /8 IPv4 proxies which are located in 2,418 ASNs and 122 countries. Also, the distribution is not only wide but also with most storage nodes residing densely in a few countries/continents. The top 10 countries (demonstrated in Table II) have notably contributed 56.1% of storage nodes and 71.3% of node IPs. And over 98% of storage nodes along with 96.4% of IPs are distributed in Europe, North America, and Asia-Pacific continents. This phenomenon may lead to inconsistent storage performance of users located in different regions. For instance, users in Asia-Pacific have an inferior strength compared to Europe or the United States, and we will verify this assumption in §IV-A of QoS.

[5]https://www.storj.io/blog/our-approach-regarding-russian-network-connectivity-uncertainty

[6]https://ngrok.com/

TABLE II
TOP 10 COUNTRIES ACCOUNTING FOR MOST STORAGE NODES.

Index	Country	# Nodes	% Nodes	# IPs	% IPs
1	United States	4,439	13.5%	9,963	6.4%
2	Germany	3,584	10.9%	70,450	45.32%
3	France	2,564	7.8%	5,149	3.31%
4	Russia	2,104	6.4%	4,919	3.16%
5	Spain	1,512	4.6%	7,283	4.68%
6	Netherlands	986	3%	3,028	1.94%
7	Canada	953	2.9%	3,257	2.1%
8	United Kingdom	855	2.6%	2,409	1.55%
9	Romania	789	2.4%	2,331	1.5%
10	Ukraine	657	2%	2,129	1.37%

Churn Rates of Storage Nodes. We also measured the churn rates of storage nodes, in an attempt to further understand the stability of the Storj network. Among storage nodes initially observed in April 2021, only 44% were still active by the last month of our measurement, namely, August 2022. We also calculated the month-by-month churn rates between May 2021 and August 2022. As listed in Table III, the churn rate ranges from 4.55% to 15.28%, and the average churn rate is 9.6%. Besides, to corroborate our findings, we further investigated the Storj disqualified nodes, and as aforementioned, 32,881 unique storage nodes are observed in total while the daily number of active storage nodes is stable at around 13K. One root cause we have identified is that many storage nodes may be considered by the satellite as disqualified after failing many audits (e.g., offline, stored data loss). Once a storage node is regarded as disqualified, it will be excluded from serving data storage in which case our collector will not be able to capture it anymore [17]. In particular, we crawled the statistics of disqualified nodes released by Storj DCS Public Network Statistics [18]. From [18], the number of disqualified storage nodes increases from 22K in June 2021 to 37K in August 2022, which aligns with our empirical observations. And such non-negligible churn rates and disqualified nodes highlight the Storj network still has a particular deficiency that needs to progress, especially in how to retain the storage node operators.

Finding 1: On the one hand, the Storj network has a large number of storage nodes that are globally distributed in 122 countries with steady growth, which highlights its distributed characteristics and suggests a high resilience. Still, there is a phenomenon that most nodes are deployed in a few countries. On the other hand, the number of daily active storage nodes is relatively stable with a 9.6% churn rate month-by-month during our 16-month measurement.

IV. QUALITY OF SERVICE

Upon a comprehensive understanding of the Storj ecosystem, we then move forward to profile its quality of service. Our performance evaluation is focused on answering the following

TABLE III
MONTH CHURN RATE COMPARED TO THE LAST MONTH.

Month	2021-05	2021-06	2021-07	2021-08	2021-09
Churn Rate	14.48% (1817)	10.93% (1401)	10.33% (1303)	9.43% (1179)	7.56% (947)
Month	2021-10	2021-11	2021-12	2022-01	2022-02
Churn Rate	13.41% (1722)	15.28% (1860)	14.47% (1558)	4.55% (515)	8.45% (1130)
Month	2022-03	2022-04	2022-05	2022-06	2022-07
Churn Rate	7.67% (1068)	8.06% (1129)	7.54% (1084)	5.29% (778)	6.21% (950)

three questions: 1) *What QoS can Storj offer?* 2) *How does the QoS of Storj vary across regions and access options?* 3) *How well is Storj's QoS when compared to major centralized cloud storage services?*

Methodology and Implementation. Our performance evaluation framework consists of an evaluation controller, the storage clients (Storj Uplink CLI and Storj S3 CLI), as well as storage servers of three cloud storage services including Storj, and 2 other representative centralized cloud storage services, Amazon S3 [2], and Google Cloud Storage [1]. The Storj Uplink CLI is a Python wrapper that we built upon the Storj Uplink. And the S3 CLI was implemented based on `Boto3`[7], the AWS S3 SDK for Python.

In our study, we considered various file sizes to upload and download. As mentioned in §II, each segment has a maximum size of 64MB, so we choose 64MB, and 128MB, as well as other three small file size settings (1MB, 4MB, and 16MB) which are commonly adopted in previous works [19]. Besides, to mitigate biases potentially incurred by geographical locations, we deployed our clients in three representative regions including Singapore in Asia-Pacific (AP-SG), Germany in Europe (EU-DE), and the United States in North America (NA-US). Also, storage servers located in these regions were considered in our experiments. Further, we throttled the network bandwidth of the storage clients to various extents, aiming to evaluate how it affects the end2end storage performance. This is favored by a bandwidth throttling tool, `WonderShaper` [20]. For each unique setting, 5 duplicate experiments were carried out to achieve more reliable performance statistics. Overall, we have carried out over 49,000 file storage operations involving different combinations of client locations, storage services, client bandwidths, file operations, and file sizes. Also, two commonly recognized QoS metrics [19], [21], latency (length of time in milliseconds that it takes for data transmission) and throughput (MBs per second), are adopted in our study to profile storage performance. Table IV details our evaluation settings.

A. The QoS of the Storj Network

Default Setting. The default setting as listed in Table IV consists of a Storj-Uplink client deployed in the EU-DE with a 100Mbps bandwidth, as well as the Storj satellite also in the EU. In the default setting, a file of 64MB is uploaded and downloaded multiple times with predefined performance metrics measured. In short, Storj-Uplink has achieved an average latency of 18.97 seconds as well as an average throughput of 3.39 MB/s for the upload operation, whereas its download performance is slightly better with 7.6s as the latency and 8.44 MB/s as the throughput.

[7]https://github.com/boto/boto3

TABLE IV
CONFIGURATIONS OF OUR PERFORMANCE EVALUATION.

Name	Options	Default
File Operations	Upload, Download	N/A
File Size in MB	1, 4, 16, 64, 128	64
Server Locations[1]	AP, EU, US	EU
Client Configuration	Ubuntu 18.04, 1 CPU, 2 GiB Memory	N/A
Client Locations	AP-SG, EU-DE, NA-US	EU-DE
Client Bandwidth in Mbps	10, 50, 100, 150, 200, 250, 300	100

[1] A client can only select the intercontinental region to which the satellite belongs, rather than more fine-grained, such as countries or individual nodes.

TABLE V
LATENCY (S) OF STORJ UPLINK IN DIFFERENT REGIONS.

	Upload			Download		
Client/Satellite[1,2]	AP	EU	US	AP	EU	US
AP-SG	**20.82**	21.23	24.80	**9.97**	12.87	15.11
EU-DE	22.39	**18.97**	23.01	8.56	**7.6**	8.47
NA-US	22.44	21.4	**19.0**	8.64	8.53	**7.58**

[1] Client bandwidth: 100Mbps, upload/download file size: 64MB
[2] The average standard deviation for upload and download is 1.7s and 0.9s, respectively.

Satellite Locations. Following Storj's design, nodes and customers are divided by regions, and the satellites are set up in each region to coordinate storage activities in the respective region. Intuitively, a client should connect to the satellite in the same region so as to achieve the best storage performance. Hence we evaluate the hypothesis through comparative experiments. Specifically, in our experiments, when conducting the default storage operations, clients are instructed to connect to satellites not only in the same region but also in the other 2 regions. Comparing the performance variations, we are allowed to learn whether it is the best strategy for a client to always connect to the same-region satellite.

Table V presents the experiment results and we can see that a client in regions AP-SG, EU-DE, and NA-US can achieve better performance when connecting to satellites in its home region. This can be explained by the fact that the satellite tends to choose storage nodes that are close to the client to serve an upload operation. In addition, we also find that a client in the EU-DE or NA-US can achieve a better performance compared to an AP-SG client (both in file upload and download operations). A good explanation is that most storage nodes are located in either Europe or North America, as above-mentioned in §III-B. In other words, the more storage nodes the home region has, the better storage performance a client may achieve.

Storj Uplink VS Storj S3 CLI. Storj offers two options for a client to access its storage network. One is Storj Uplink, which communicates directly with storage nodes with help from satellites, the other is a general S3 client which communicates only with a Storj S3 gateway server [11], which in turn handles interactions with Storj nodes and satellites. It is interesting

TABLE VI
THROUGHPUT (MB/S) COMPARISON BETWEEN STORJ UPLINK AND S3.

Configuration[1]	1MB	4MB	16MB	64MB	128MB
Uplink, Upload	0.51	1.71	2.99	3.39	3.49
S3, Upload	**0.63**	**2.17**	**7.05**	**10.32**	**10.8**
Uplink, Download	1.27	3.53	8.47	8.44	8.59
S3, Download	**2.01**	**5.39**	**14.66**	**11.85**	**11.12**

[1] Clients in EU-DE, and client-side bandwidth is 100Mbps.

to explore which access option can provide better storage performance. We thus conducted a set of experiments to answer these questions. Based on the results presented in Table VI, it is apparent that the Storj S3 gateway exhibits superior performance in pre-defined scenarios. Also, regardless of the client options, download operations have achieved better throughput compared to respective uploads. To explain the aforementioned performance observations that the Storj S3 provides such a better performance, we carried out a theoretical analysis in detail.

Theoretical Explanation. Given the performance comparison results between Storj Uplink and Storj S3 (mentioned in Table VI), we provide reasonable explanations regarding why the S3 client can achieve better results compared to Uplink, and also what makes downloading have a higher throughput than uploading. We assume the client-side bandwidth is b_c Mbps whereas it is b_{s3} Mbps for the Storj S3 gateway server. Also, the erasure code parameters are set up as $(k, l, n, o) = (29, 39, 80, 110)$ in the current Storj implementations, which may change in the future. As introduced in §II, Storj Uplink will divide an uploaded file into 64MB segments. Here we denote the file size as x MB (less than 64MB), and the erasure encodes each segment into n pieces with each of the size of x/k before uploading these pieces to o storage nodes (if the long-tail issue arises,). Let's assume the storage nodes have enormous bandwidth and a file upload/download operation is mainly throttled by the client-side bandwidth. That being said, the upload latency of Storj Uplink L_{U-U} can be represented as Equation 1a.

Different from using Storj Uplink, file upload through Storj S3 gateways involves two steps, one is to upload the whole file to the gateway server, then the gateway can serve as the Storj Uplink and be used to upload the erasure-coded pieces to individual Storj nodes. Therefore, as detailed in Equation 1b, its upload latency L_{S3-U} can be calculated as $\frac{8x}{b_c} + \frac{8xo}{b_{s3}k}$. In this analysis, we focus solely on the impact of client or S3 gateway bandwidth on latency and throughput, excluding commonly shared latency-relevant factors such as storage node bandwidth and possible packet loss. If we want $L_{U-U} < L_{S3-U}$ (i.e., having Storj Uplink perform better than Storj S3), b_c has to satisfy that $b_c > \frac{o-k}{o} b_{s3}$, and vice versa. We can see that performance of Storj Uplink can be better than Storj S3 as long as the client-side upload bandwidth approximates or even exceeds the bandwidth available on the Storj S3 gateway. Also, the download latency for Storj Uplink and S3 clients are formalized respectively in Equation 1c and Equation 1d. Since $l < o$ in the current Storj implementation, a download operation incurs less traffic and thus has a lower latency when compared to the respective upload operation.

$$L_{U-U} = \frac{8x/k}{b_c/o} = \frac{8xo}{b_c k} \quad (1a)$$

$$L_{U-D} = \frac{8xl/k}{b_c} = \frac{8xl}{b_c k} \quad (1c)$$

$$L_{S3-U} = \frac{8x}{b_c} + \frac{8x/k}{b_{s3}/o} = \frac{8x}{b_c} + \frac{8xo}{b_{s3}k} \quad (1b)$$

$$L_{S3-D} = \frac{8xl/k}{b_{s3}} + \frac{8x}{b_c} = \frac{8xl}{b_{s3}k} + \frac{8x}{b_c} \quad (1d)$$

Case Study. Here we provide a case study to further demonstrate the aforementioned theoretical analysis. Specifically, in an upload operation, consider the file size is 64 MB ($x = 64$), and the EU-DE client-side bandwidth is 100Mbps ($b_c = 100$). Also, erasure coding parameters are $k = 29$ and $o = 110$ as aforementioned. Leveraging Equation 1a, we can get $L_{U-U} \approx 19$s of Uplink in theory which is found to be consistent with our realistic experiment results shown in Fig. 2. On the other hand, an S3 gateway server can have a bandwidth of multiple Gbps. Here we assume the gateway bandwidth as 1,000 Mbps ($b_{s3} = 1000$). In this case, $L_{S3-U} \approx 7$s which is consistent with Fig. 2.

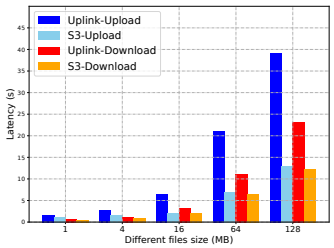
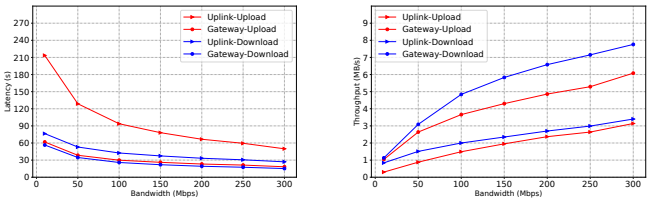


Fig. 2. Latency performance with different access options between Storj S3 gateway and Storj Uplink.

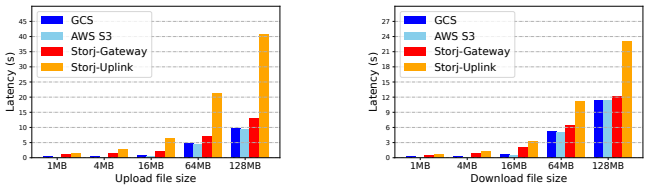
In short, we have carried out a set of case studies that have verified the effectiveness of the proposed equations in terms of approximating real-world latency. The occurrence of this phenomenon is not solely influenced by the Reed-Solomon erasure codes [13] implemented in the Storj network, but also by the bandwidth on the client-side. In order to have the Uplink client achieve a performance comparable to or even better than the S3 client, the client-side bandwidth should be approximated to that of the Storj S3 gateway server. To verify this hypothesis, next we will explore the client-side network bandwidth with a focus on profiling its effect on storage performance.

Client Bandwidth. We set the client access options as Storj Uplink and S3 CLIs, and the client region as AP-SG because the AP-SG client-side cloud service can provide a larger network bandwidth compared to other regions. Then we throttled the client machine to 7 different bandwidth values (10Mbps to 300Mbps as shown in Table IV) and instructed the client to conduct both upload and download operations. Fig. 3 presents the throughput and latency fluctuations achieved by clients with different network bandwidth settings, wherein each line denotes a combination of a file operation upon a file of 64 MB. We observed that with the network bandwidth increasing, the latency of file operations gradually decreases regardless of the



(a) Latency. (b) Throughput.

Fig. 3. The performance between Storj S3 gateway and Storj Uplink with bandwidth throttling.



(a) Upload. (b) Download.

Fig. 4. Latency of all four storage options when handling files of various sizes (Location: EU-DE, Bandwidth: 100Mbps).

client types, which is proving the correctness of the assumption put forward in the *Theoretical Explanation* part.

> **Finding 2:** As introduced above, we have conducted a set of control experiments to profile Storj's performance in different configurations. It turns out that customers in EU-DE or NA-US can enjoy a better performance compared to AP-SG, due to the unbalanced availability of storage nodes in different regions. And the Storj S3 is a preferable CLI to upload a file compared to the Storj Uplink in terms of performance.

B. Comparison with Centralized Cloud Storage

In this part, we move forward to compare Storj's performance with two representative centralized cloud object storage services, in an attempt to understand its potential to complement or even replace centralized cloud storage services. To profile their performance differences, we designed and carried out a set of controlled experiments, e.g., various network and computing settings, different file operations, and comparisons of different storage services.

File Sizes. For a more comprehensive comparison between decentralized and centralized object cloud storage systems, we explore to what extent storage services can perform differently when handling files of different sizes. Fig. 4 presents the latency results of four storage options, when uploading or downloading files of various sizes, in region EU-DE with client-size bandwidth of 100Mbps. And we can see that if the size of the file is less than 64MB, AWS S3 [2] and GCS [1] can gain much better performance than Storj. However, as the file size increases, the performance of Storj S3 is progressively getting closer to the two centralized ones. Note that this observation is also found consistent across other combinations of client-side bandwidth and regions.

Client Locations. Similar to Storj, AWS S3 and GCS also have server endpoints in AP, EU, and US. We thus step forward to evaluate the potential performance variance across regions for these storage options. Still, controlled experiments are conducted with each involving a different region. In each controlled experiment, clients are instructed to carry out file operations against storage endpoints located in the same region. Latency results are listed in Table VII for a file of 64MB and a client bandwidth of 100Mbps. Regardless of upload or download, AWS S3 and GCS are found to have achieved better performance compared to Storj Uplink and Storj S3. However, in some scenarios, such as a client uploading data in the NA-US or a client downloading data in the EU-DE, the Storj S3 can also achieve a performance that is comparable to GCS. But in any case, Storj Uplink still has the worst performance in data storage, among the four options.

TABLE VII
LATENCY (S) RESULTS MEASURED AGAINST DIFFERENT REGIONS WITH A FILE OF 64MB AND A CLIENT-SIDE BANDWIDTH OF 100MBPS.

Region	Upload				Download			
	Uplink	S3[1]	GCS[1]	AWS[1]	Uplink	S3	GCS	AWS
AP-SG	20.83	8.78	6.41	4.52	9.97	6.64	5.30	4.93
EU-DE	18.97	6.31	4.75	4.48	7.60	5.41	5.14	4.96
NA-US	19.70	5.91	5.48	4.42	7.58	5.87	4.96	4.84

[1] S3 denotes Storj S3, GCS and AWS stands for Google Cloud Storage and AWS S3.
[2] The average standard deviations of upload and download for Uplink, S3, GCS, and AWS are 1.7s, 0.86s, 0.5s, 0.7s and 0.4s, 0.5s, 0.23s, 0.75s, respectively.

Pricing Comparison between Storj and CCS. Nevertheless, when a user selects a cloud storage option, pricing is another important factor. We thus also investigated the pricing policies of Storj [22] and the two traditionally centralized cloud storage systems (AWS S3 pricing [23] and GCS pricing [24]). In a nutshell, the storage cost depends on two key factors, one is the size of the stored data, and the other is the bandwidth consumption of data retrieval. We compared Storj with the two centralized storage options in Fig. 5 which plot the monthly cost of storing data across storage options, data size, and download bandwidth consumption. For instance, when storing 100GB of data with 1TB download bandwidth consumption, it will cost 94.6 dollars in AWS S3, 124.9 dollars in GCS, and only 10.7 dollars in Storj. In summary, Storj provides a more economic storage option compared to GCS and AWS S3, regardless of the storage volume and bandwidth consumption.

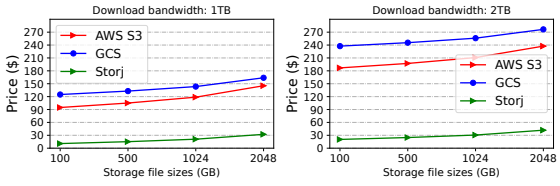
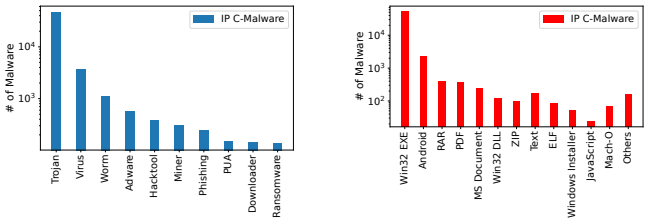


Fig. 5. Pricing comparison between Storj and CCS services in NA-US.

Finding 3: As profiled above, Storj still falls behind in performance compared to traditional CCS systems, especially in small file transmission. The root cause for this is not only associated with the erasure code redundancy mechanism but also influenced by the quantity of storage nodes in the region. When it comes to pricing, which is a crucial factor for most customers in choosing a cloud storage provider, Storj holds a significant advantage over centralized services.



(a) Malware categories. (b) Malware file types.

Fig. 6. Distribution of communicating malware samples across malicious categories and file types.

V. Security

In this section, we further profile the security implications of the Storj network. In §V-A, we first evaluate the network threats co-located with Storj storage nodes, i.e., the extent to which Storj nodes (IPs) have been involved in malicious activities. We then look into the Storj protocols and have identified a security vulnerability that allows a Byzantine node to manipulate its storage reputation (also called reputation manipulation vulnerability) in §V-B.

A. Threat Reputation of Storage Nodes

Various security measures have been deployed to detect and block malicious network traffic, such as firewalls and deep packet inspectors. The reputation of IP addresses involved in a traffic flow is a critical factor well considered by these security tools to decide the actions. A traffic flow is more likely to be blocked when it involves IP addresses that are either heavily abused or under the control of miscreants [25]. In this part, we profile the network reputation for the 155,457 IP addresses of storage nodes by VirusTotal [26], which is a well-recognized open threat intelligence to extract threat reports. From the observation, 4.48% of all the storage nodes' IPs have been associated with at least one malicious activity. The associations with malicious activities can be further divided into two categories including communicating with malware samples (5,643 IPs, 3.63%) such as being C2 servers in botnets, and referred to or embedded as part of malware payloads (1,321 IPs, 0.85%).

Communicating and Referrer Malware. We further look into the 5,643 storage IP addresses that have been contacted by one or more malware samples. First of all, 19,202 unique malware samples have ever communicated with these IPs in total. And these IPs have ever hosted 4,869 (14.8%) storage nodes. Leveraging the malware metadata included in VirusTotal reports, we are allowed to group these communicating malware samples (C-Malware) by their categories, the top 10 of which are listed in Fig. 6(a). Specifically, over 86% of malware samples belong to Trojan while 6.7% are labelled as Virus. Particularly, 0.3% of malware samples are crypto mining software, which suggests some storage nodes are co-hosted on the same IPs with suspicious mining activities. For instance, "5.*.*.178" used to serve as a storage node ("1Jkrxg****J7BA") as well as a mining pool server (*litecoinpool.org*) at the same time. In addition, Fig. 6(b) presents the distribution of malware samples in terms of their file types, around 76% are Win32 EXEs while 20% are Android apps.

Besides, we also shed light on referrer malware samples, i.e., malware samples with each embedding in its payload at least one storage IP address. Statistically, over 1,321 storage node IP addresses have been referred to by 1,050 unique malware samples. It is surprising that some of the IP addresses referred to by the malware are private IP addresses. After further excavation of this discovery, we found that over 102 private IP addresses correspond to 107 unique storage nodes. This implies that the private IPs stored in satellites are not reachable to Storj storage nodes, which can disrupt the data uploading process of storage clients. We also ordered the top 10 referrer malware samples by the number of storage IPs each of them has referred to. It's very interesting that the malware samples named "Perfect-Privacy-VPN_Setup.exe", which is a port forwarding service provider (*Perfect Privacy* [27]), have referred to the most storage node IPs associated with over hundreds of storage nodes. This suggests commercial port forwarding services are also utilized by storage nodes to either hide their real public IP address or screen their internal storage servers to the public, as depicted in §III-B. We reported the above-mentioned observation to Storj.io, and the Storj team evaluated it and concluded it was a common threat.

Port Forwarding Services. To investigate this observation, we studied the public port forwarding services adopted by storage nodes in the Storj network. Among the 155K storage IPs, 1,492 (0.96%) are found to be exclusively used to serve as the tunnel servers of port forwarding services. The list of the involved port forwarding services such as *pktriot.net, packetriot.net*, *portmap.host*, *portmap.io, airvpn.org*, *airdns.org*, and *ngrok.io*. Despite allowing running a storage node even without a public IP address, port forwarding services can compromise the reputation of storage nodes since such services have been found to be abused by various malicious activities such as botnets and spam distribution, as revealed in VirusTotal reports and previous works [28]. Therefore, co-location with malicious activities on the same IP will likely render a storage node suspicious, which can further get traffic toward a storage node blocked by security tools such as firewalls. For instance, a seriously polluted storage IP, "193.*.*.99", can in the meanwhile host tens of storage nodes. Further analysis reveals that this IP address is used

to serve as a C2 server for multiple botnets and is actually exclusively used by a port forwarding service, *portmap.io*, to host tunnels toward network services deployed in private networks. Apparently, these port forwarding services have been used for tunneling traffic toward both storage nodes and various malicious servers (e.g., C2 servers in botnets).

> **Finding 4:** Many storage IP addresses (4.48%) are observed to have been associated with various malware samples. Additionally, internet-wide port forwarding services are widely used by storage nodes (0.96% storage nodes IPs), which can make storage nodes co-locate with various malicious activities such as botnets and cryptomining. This can further compromise the reputation of involved storage nodes and thus undermine the availability and resilience of the Storj.

B. Storage Reputation Manipulation

In this part, we present another fundamental security risk or vulnerability that is inherent in Storj's design. In a nutshell, a Byzantine storage node, especially a node with an unauthentic low reputation, is found to be able to maliciously tamper its storage reputation in an attempt to improve its chance of being selected for data storage.

Reputation in Storj. As mentioned in the background section (§II), a storage node is responsible for data storage and data retrieval and earns rewards in return. From [29], the more frequently that a storage node is selected to store new data, the more egress bandwidth it can likely incur, and the more rewards the storage node can receive. Every time a new segment is uploaded to Storj, the satellite will carry out a node selection process to surface out $n = 110$ from the pool of tens of thousands of storage nodes, so as to store the erasure-coded pieces. Nodes are selected by considering a set of factors (e.g., subnet, latency, reputation), among which, reputation is one of the most important factors [30]. A storage node with a high reputation tends to have a higher chance of being selected by the satellite. And during the node selection process, various reputation factors will be considered. These reputation factors include audit failures, successful PUT/GET count, PUT/GET selection count, PUT/GET selection success count, etc. Specifically, the longer a storage node joins the Storj network, the higher its PUT selection count, which will thus have a positive impact on its reputation.

Nevertheless, our analysis of Storj's source code reveals that some reputation factors can be easily manipulated. In other words, a Byzantine node can forge a set of reputation factors (PUT/GET selection count, successful PUT/GET, and PUT/GET selection success count, etc) to improve its overall reputation. We consider such kinds of misbehavior as a reputation manipulation vulnerability.

Testbed. To demonstrate the reputational manipulation risks described above, we deploy the Storj (v1.40.4[8]) Test Network [31] on a Linux server (Ubuntu 20.04) with 256GB RAM as a testbed. Our testbed consists of one Uplink client, ten unique storage nodes, and one satellite. Leveraging this testbed, we evaluate multiple feasible vulnerability channels for a Byzantine storage node to manipulate reputation factors and intentionally enhance its overall reputation.

Implications. An important security vulnerability in Storj is the collusion between a Byzantine storage node and a storage client. Specifically, an adversary can serve as a storage node and a storage client at the same time, and modify the Uplink client so that it can be instructed to upload file pieces to the designated storage nodes (under the adversary's control) rather than the ones returned from the satellite. Leveraging this technique, a Byzantine storage node can quickly improve its reputation. In addition to promoting a Byzantine storage node, an adversary can also demote an authentic storage node by forging upload/download failures. Such failures will lower the success rates of PUT and GET operations, and thus compromise the reputation of a victim storage node.

To demonstrate this vulnerability, we conduct experiments in the testbed leveraging a modified Uplink CLI and confirm that the reputation factors of the corresponding storage nodes can be successfully manipulated. As a result, unauthentic storage nodes may gain unfair advantages over authentic nodes when competing for storage requests, and data can get stored in low-reputation nodes which can incur a higher probability of data loss. Also, authentic storage nodes will mistakenly receive fewer incentives and have a higher probability of dropping out of the network. In short, storage reputation manipulation will compromise the stability and scale of the Storj network. We have also reported this responsible disclosure to Storj.io, and Storj's team assessed this to be a common vulnerability.

> **Finding 5:** As discussed above, important reputation factors can be manipulated by an attacker by forging storage transactions. Particularly, malicious storage nodes can unfairly optimize their reputation, while compromising that of an authentic storage node, which may disrupt the Storj ecosystem to some extent.

VI. Ethics

In this study, we take ethical issues seriously and have carefully designed our methodologies in a way that can avoid ethical concerns. First, the proposed storage node collector does not interact with storage node operators and customers of the Storj network. Instead, the collector only sends requests to the satellites at a very limited rate. We run the node collector on a daily basis with a fairly low request frequency (as mentioned in §III-A), and we believe our collection jobs should not have disrupted the regular operations of the satellites. Besides, as aforementioned in §IV, we uploaded large-scale files to the Storj network with payments, and we believe that these file operations (upload/download) have no ethical implications for the ecosystem. Lastly, when exploring the security vulnerabilities of the Storj network, we evaluated

[8]https://github.com/storj/storj/releases/tag/v1.40.4

the vulnerabilities in a controlled and simulated environment, rather than the realistic Storj network.

VII. Related Work

Decentralized Cloud Storage. A set of survey studies [4], [8], [21] are there to provide an overview of DCS services from various aspects. Nazanin et al. [8] conduct a survey on four blockchain-based decentralized storage systems, namely, Storj, Sia, Filecoin, and Swarm[9]. Similarly, Daniel et al. [4] give a qualitative comparison of these emerging and decentralized cloud storage services, with future challenges derived and research goals distilled. Also, [10] is dedicated to studying IPFS's performance, security, and open issues. Different from these studies on DCS, our work carries out the first empirical study on Storj, and our performance measurement is conducted on realistic storage systems rather than simulated ones.

Research on Storj. Zhang et al. [32] proposed an attack, namely, the frameup attack, in which the attacker can store unencrypted data on the hard drives of a victim which shares the hard drives for rewards, and thus open the door for framing the victim. And the authors utilized Storj as a testbed to demonstrate the proposed attacks. Furthermore, Figueiredo et al. [9] provide an architectural overview of Storj along with the discovery of a DoS vulnerability. Different from these works, we carry out an extensive and empirical measurement of Storj, along with novel findings regarding its landscape, quality of service, and security implications.

VIII. Conclusions

We present the first empirical measurement of Storj, with a focus on its ecosystem, performance, and security implications. Our research is enabled by novel measurement techniques to capture storage nodes, profile Storj's quality of service, and evaluate the potential risks in a simulation environment. A set of novel findings are distilled, particularly, a large volume of storage nodes are observed with a wide distribution. However, Storj customers in different regions can have a different QoS likely due to the imbalanced distribution of storage nodes. Also, among IP addresses hosting storage nodes, 4.48% have been associated with various malicious activities (e.g., botnets, cryptomining), which can likely compromise the reliability of the Storj network. Besides, we also have discovered vulnerabilities in Storj's auditing system where Byzantine nodes could maliciously boost their reputation and demote competing nodes, which can potentially undermine the quality of service and the fairness of the Storj network.

Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments. This project is jointly supported by USTC (The Innovation Fund for Young Investigators), MSRA (Research Awards for Responsible AI), Shandong Provincial Natural Science Foundation (No. ZR2020MF055, No. ZR2021LZH007, No. ZR2022LZH013, No. ZR2020LZH002, No. ZR2020QF045), and National Natural Science Foundation of China under Grant No. 62002203.

[9]https://www.ethswarm.org/

References

[1] Google Inc., "GCS," https://cloud.google.com/storage/, 2022.
[2] Amazon Inc., "Amazon s3," https://aws.amazon.com/s3/, 2022.
[3] "Google services outages," https://en.wikipedia.org/wiki/Google_services_outages#August_2020_services_outage, 2022.
[4] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks," *arXiv preprint arXiv:2102.12737*, 2021.
[5] Storj Labs Inc., https://www.storj.io/, 2022.
[6] Sia, https://sia.tech/, 2022.
[7] "IPFS and FileCoin," https://docs.filecoin.io/about-filecoin/ipfs-and-filecoin/, 2022.
[8] N. Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *J. Netw. Comput. Appl.*, vol. 162, p. 102656, 2020.
[9] S. de Figueiredo, A. Madhusudan, V. Reniers, S. Nikova, and B. Preneel, "Exploring the storj network: a security analysis," in *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*. ACM, 2021, pp. 257–264.
[10] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and evaluation of IPFS," in *Proceedings of the ACM SIGCOMM 2022 Conference*. ACM, aug 2022.
[11] Storj.io, "Storj-hosted s3 compatible gateway," https://docs.storj.io/dcs/api-reference/s3-compatible-gateway/, 2022.
[12] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. ACM, 2007, pp. 584–597.
[13] I. S. Reed and G. Solomon., "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics, 8(2):300–304*, 1960.
[14] "Filecoin FAQ," https://docs.filecoin.io/about-filecoin/faq, 2022.
[15] "Free tier," https://docs.storj.io/dcs/concepts/limits/#free-tier, 2022.
[16] IPinfo, https://ipinfo.io/, 2022.
[17] S. N. Support, "Why is my node disqualified?" https://support.storj.io/hc/en-us/articles/4403035941780-Why-is-my-node-disqualified-, 2021.
[18] "Storj DCS public network statistics," https://stats.storjshare.io/, 2022.
[19] J. Shen, Y. Li, Y. Zhou, and X. Wang, "Understanding I/O performance of IPFS storage: a client's perspective," in *Proceedings of the International Symposium on Quality of Service, IWQoS 2019, Phoenix, AZ, USA, June 24-25, 2019*. ACM, 2019, pp. 17:1–17:10.
[20] B. Hubert, J. Geu, and S. Séhier, "The wonder shaper," https://github.com/magnific0/wondershaper, 2021.
[21] O. F. Cangir, O. Cankur, and A. Ozsoy, "A taxonomy for blockchain based distributed storage technologies," *Information Processing & Management*, vol. 58, no. 5, p. 102627, 2021.
[22] "Stoj pricing," https://docs.storj.io/dcs/billing-payment-and-accounts-1/pricing, 2022.
[23] "Amazon s3 pricing," https://aws.amazon.com/s3/pricing/, 2022.
[24] "Cloud storage pricing," https://cloud.google.com/storage/pricing, 2022.
[25] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, 2016, pp. 807–822.
[26] VirusTotal, https://www.virustotal.com/, 2022.
[27] "Perfect privacy vpn," https://www.perfect-privacy.com/en/, 2022.
[28] S. Tang, X. Mi, Y. Li, X. Wang, and K. Chen, "Clues in tweets: Twitter-guided discovery and analysis of SMS spam," *CoRR*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.01233
[29] J. Gleeson, "Sharing storage space for fun and profit," https://www.storj.io/blog/sharing-storage-space-for-fun-and-profit, 2019.
[30] ——, "Reputation matters when it comes to storage nodes," https://storj.io/blog/reputation-matters-when-it-comes-to-storage-nodes, 2019.
[31] S. L. Inc., "Storj test network," https://github.com/storj/storj/wiki/Test-network, 2021.
[32] X. Zhang, J. Grannis, I. M. Baggili, and N. L. Beebe, "Frameup: An incriminatory attack on storj: A peer to peer blockchain enabled distributed storage system," *Digit. Investig.*, vol. 29, pp. 28–42, 2019.