The State of Privacy in IPFS

Patrick Kühtreiber[1] **and Cornelius Ihle**[1]

[1]**University of Göttingen**

Corresponding author:
Patrick Kühtreiber

Email address: kuehtreiber@cs.uni-goettingen.de

ABSTRACT

InterPlanetary Filesystem (IPFS) is a permissionless set of peer-to-peer protocols. Its main goal is to provide a distributed alternative to HTTP. However, IPFS entails substantial privacy risks, as content requests and offerings are openly shared and not obscured. That means that no one within the IPFS network is anonymous, nor can sensitive content be offered privately. We review scientific papers, talks at IPFS conferences, and blog articles to give a thorough overview of privacy improvements for content requests and offerings. Our results indicate that complete anonymity is impossible within the IPFS network and that the most promising mitigations revolve around the privacy goal of *plausible deniability*. However, future research could investigate incorporating anonymity networks, such as Tor, inside IPFS to enable provider anonymity.

INTRODUCTION

In HTTP, clients use the location of a file to request it from a server. This approach has limitations when distributing datasets of petabyte proportions or when versioning is needed. To combat these limitations, technologies such as Git and BitTorrent have been developed.

Benet (2014) revisited the problem with a strong focus on decentralization. He proposed a universal file system—the IPFS. IPFS is a Peer-to-Peer (P2P) network in which clients directly ask for content using its unique self-validating Content Identifier (CID)—the file's cryptographic hash (Benet, 2014). Thus, clients will always verifiably get exactly the content they are asking for since the CID will change if the file has been altered. This way, IPFS provides data integrity—a primary protection goal of the CIA-triad framework (Samonas and Coss, 2014), which is widely used to develop security policies that focus on confidentiality, integrity, and availability.

While solutions to also protect the confidentiality of the data exist and the system's design inherently grants availability, privacy is not an integral part of IPFS. In fact, the official documentation states that due to its modular design, developers should take care of privacy themselves (IPFS Docs, 2023e). Moreover, research on privacy methods tailored to IPFS is scarce (Daniel and Tschorsch, 2023), and commercial and open-source solutions are hard to come by. However, privacy problems are plentiful when using IPFS and include content privacy, client- and provider anonymity, and user tracking (Matt Ober, 2019; Tourani et al., 2017; Chaabane et al., 2013). These problems are also concerning for current users[1]. IPFS has been proposed as a viable solution for multi-party collaboration (Nizamuddin et al., 2019), scientific publishing and peer review (Tenorio-Fornés et al., 2019), and social networking (Xu et al., 2018).
In this paper we aim to provide a comprehensive overview, comparison, and classification of the privacy solutions available for IPFS. This review is intended for the researchers, privacy scholars, and developers who are working with or studying IPFS and other decentralized storage protocols. It also targets security and privacy professionals who seek to understand IPFS from a privacy-focused perspective and who may contribute to the development of tools and protocols to mitigate privacy risks.

In detail, we investigate the following research questions:

- RQ1 What are the key privacy issues when using IPFS for a) users and b) providers?
- RQ2 Which solutions exist to address privacy issues for a) users and b) content providers?

[1] https://www.reddit.com/r/ipfs/comments/kpicby/

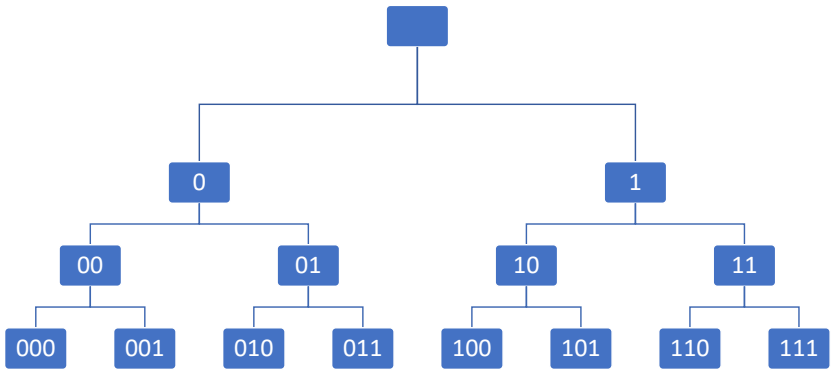

Figure 1. Kademlia tree in the 3-bit keyspace

BACKGROUNDS

In this section, we present background information on IPFS, its data-exchange protocol Bitswap, HTTP gateways, privacy, and finally, Private Information Retrieval (PIR).

IPFS

IPFS is a suit of P2P protocols intended to change the way data is stored and addressed. IPFS was designed by Protocol Labs[2] and is now maintained by IPShipyard[3]. The data is distributed over a P2P network in which each node functions as both a client and a server, with each role bearing different privacy challenges. This decentralized approach eliminates single points of failure and prevents censorship. If a client wishes to fetch a file, they directly address it using its CID instead of its location. Files on IPFS get split into blocks of 256 KB—a process called "chunking" (Carson Farmer, 2018). Each chunk gets a separate CID, and all chunks are connected to a tree-like structure called the Merkle Directed Acyclic Graph (DAG) (IPFS Docs, 2023d). The CID used to access the original file is, in fact, the root CID of the respective Merkle DAG (IPFS Docs, 2022). A key property of the Merkle DAG is its self-verifying structure. By changing one node (i.e., one block of the file) within the graph, all parent nodes change—including the root—and, thus, the original file's CID. Consequently, if the content of the file changes, so would its hash and identifier. Therefore, nodes can verify each file's integrity using CIDs.

The content is, however, not directly provided to the network. Nodes advertise that they host a piece of content by writing a pointer to themselves on the Distributed Hash Table (DHT). In IPFS, the DHT implementation is based on the Kademlia algorithm, which allows for efficient XOR metric-based routing. A so-called provider record is a key-value pair that connects the pointer (the key) to the respective file-hosting peers (values). This process is called pinning. Simply put, the CID is hashed again to obtain a binary string representing the pointer. The resulting Kademlia tree can be pictured as a binary tree that is used to lookup content along the shared prefixes using XOR-distances (see Fig. 1). With regards to privacy, it is worth noting that only if the specific pointer (the CID) is known, a node can find out the PeerID of a node providing this specific content by querying the DHT. By design, nodes cannot hide any content they hold if they wish to participate in the IPFS network—a clear privacy problem. It is, however, highly difficult to discover all content a peer is providing. From a privacy perspective, it is noteworthy that the nodes making up the IPFS network can be enumerated (Henningsen et al., 2020a,b).

Another significant advantage of IPFS is the possibility for offline access once a file is in a node's cache. An important concept regarding pinning and the cache is the garbage collector. It regularly deletes unpinned cached data (Politou et al., 2020). Pinning content ensures that it is not removed and is available within IPFS. The more nodes pin the same content, the easier and faster other nodes can find it.

Additionally, clients who own a data center may use Protocol Labs' incentive network, Filecoin (Filecoin Docs, 2023), to earn financial rewards for providing storage, proving the availability of files, and sharing them. However, its use within IPFS is not mandatory, and other incentive structures are also used (IPFS Docs, 2023b).

[2]https://protocol.ai/
[3]https://ipshipyard.com

Bitswap

The Bitswap protocol handles all file exchange requests within the IPFS network (IPFS Docs, 2023a). A file exchange is performed in three steps: (1) want lists, (2) discovery, and (3) transfer.

Nodes exchange and remember *want lists* of connected nodes. These lists contain all the CIDs of the file chunks that together make out the content. Each time a node receives a new block, it forwards the block if others want it.

During *discovery*, the requesting node sends out the root CID of wanted content to all connected peers. If none of them have the root block, the DHT is used to find nodes that do. Now that providing nodes are known, the individual wanted blocks are requested and *transferred*. Multiple nodes send different chunks in order to speed up the process (Dirk McCormick and Edgar Lee, 2020). While the transfer of data is encrypted (IPFS Docs, 2023e), it is not possible to hide which data a node requests and individual users' data requests could be traced by passive monitoring of the network (Balduf et al., 2022).

HTTP Gateways

Clients not supporting IPFS can use gateways to access IPFS via HTTP. Gateways run IPFS themselves and allow HTTP requests for IPFS-content retrieval (IPFS Docs, 2023c). If the gateway does not have the wanted file in its own cache, the request is forwarded as described in the previous section. The reliability of IPFS gateways does vary, and sometimes content cannot be retrieved (Daniel Norman, 2022). Using a gateway prevents the user's IP from getting linked to the requested content; however, the gateway used can still monitor and track the activity. Thus, trust is only shifted from the IPFS network to the gateway provider.

Privacy

A commonly accepted notion of privacy has been defined by Westin (1967): "the claim of individuals [. . .] to determine for themselves when, how, and to what extent information about them is communicated to others." This means that people should be able to determine for themselves what information to reveal to whom. In the context of IPFS, this is often referred to as PIR (see next section). Since IPFS uses a public P2P network, ensuring privacy would mean that a peer is enabled to access and offer data privately, i.e., without anyone being able to monitor requests and content transfers.

To achieve privacy, users, developers, and big companies employ Privacy-Enhancing Technologies (PETs). PETs are one or multiple methods used to preserve an individual's privacy (Gan et al., 2019). Amongst policy and filtering processes, these methods include anonymization and encryption (Shen and Pearson, 2011). Common PETs include anti-tracking technologies, anonymization tools, data perturbation mechanisms, and secure communication protocols using encryption (Kaaniche et al., 2020). Examples of privacy violations in IPFS include activity tracking, profiling, and censoring.

Private Information Retrieval

PIR allows fetching content from multiple untrusted servers without the servers learning which content was accessed. There are two variants of PIR: computational (CPIR) and information-theoretic (IT-PIR).

IT-PIR is used more commonly due to its lower computational complexity, but depends on multiple non-colluding servers. Servers holding a shared database are presented different, seemingly random query vectors from a client. The client then reconstructs the desired item from the noisy response data sets. If the servers choose to share the random vectors with each other, they are able to calculate the desired item.

CPIR on the other hand, relies on computational overhead and, therefore, does not depend on non-colluding servers. In fact, CPIR is computed on a single server. The CPIR scheme of Kushilevitz and Ostrovsky (1997) leverages number-theoretic properties to ensure retrieval privacy, i.e., a modulus N based on two large prime numbers and a set of seemingly random numbers — random quadratic residues and a single quadratic non-residue. The server calculates a response based on these seemingly random numbers. The desired data is then extracted, with the help of the quadratic non-residue, as the client knows the factorization of N. For perfect privacy, the response time in CPIR needs to be linear to the database size, becoming the main disadvantage of this original CPIR approach. By now, many Homomorphic Encryption-based solutions exist[4] that have replaced the original design. Moreover, solutions with sub-linear performance were introduced that leverage batching (Angel et al., 2018) or database encryption[5].

[4] https://blintzbase.com/posts/pir-and-fhe-from-scratch/

[5] https://spiralwiki.com

SURVEY METHODOLOGY

To answer the research questions laid out in the introduction, we first retrieve scientific papers using Google Scholar and projects (commercial and open-source) using a regular web search engine, i.e., we used a combination of the keywords *IPFS, privacy, anonymity, and plausible deniability*. We consider sources *relevant* if they address privacy problems that are specific and inherent to IPFS.

After identifying papers and other sources, we proceed by checking all references within those papers (backward search), evaluate the relevance of papers citing the original paper (snowballing by using the "cited by" option on Google Scholar), and checking all published papers by the respective author. Regarding the commercial and open-source projects that (promise to) provide a solution to IPFS privacy issues, we use various search queries containing the same keywords and check pertinent forums, such as Stack Overflow[6], Reddit[7], and IPFS Boards[8]. Since most IPFS efforts and ideas are not published but only discussed at meetings, such as IPFS Camp[9] and IPFS þing[10], we also search the IPFS YouTube channel[11] to find further presentations and talks on the subject.

RESULTS

In this section, we first present IPFS privacy issues, thus answering **RQ1**. Next, we answer **RQ2** by elaborating our findings to mitigate these issues, classified by *users* and *providers*. Note that these results are discussed in the succeeding section.

Privacy Issues

In **RQ1**, we ask: *What are the key privacy issues when using IPFS for a) users and b) providers?* The notion of privacy introduced in the previous section can not be achieved following the Bitswap protocol. If IPFS users want to fetch content, they have to broadcast the request to all connected nodes using Bitswap, and in a DHT search, the request is propagated to numerous other nodes. Hence, users cannot control who is able to monitor their request for information. While content at transit is encrypted (IPFS Docs, 2023e), IPFS was not created with privacy in mind. It was designed to be a public P2P network that relies on nodes publishing the content they want to retrieve *and* the content they possess so that other nodes can find it. This is because content is addressed via its name and not via its location (Matt Ober, 2019). Projects such as Veilid[12] aim for anonymous content-based networking by combining IPFS and Tor concepts but sacrifice interoperability with IPFS. During content retrieval, nodes broadcast their unique identifiers (*PeerID*). It is possible to connect that PeerID to a node's IP address via a DHT lookup (IPFS Docs, 2023e); therefore, anonymity for both users and content providers is not preserved. Moreover, the requested CID is also publicly available, meaning that not only does an attacker know *who* wants to fetch something but also *what* the peer wants (IPFS Docs, 2023e). Furthermore, the PeerID usually does not change, which allows tracking a user by collecting multiple requests. It is possible to provide encrypted content when specific readers have a decryption key, but this way, the content is no longer publicly consumable, and its confidentiality depends on the secrecy of the key.

Balduf et al. (2022) classified privacy threats into three attacks:

1. *Identification*: Find out which PeerIDs are interested in which CIDs via content monitoring.
2. *Tracking*: Since content requests are broadcast to all connected peers, establishing and keeping a connection to the victim is sufficient to track all its requests.
3. *Past interests*: Abusing the fact that nodes cache previously requested content, an attacker can test whether the victim has recently requested a specific CID by demanding it themselves from the user.

Monitoring projects confirm such attacks' feasibility (Balduf et al., 2022).

[6] https://stackoverflow.com/
[7] https://www.reddit.com/r/ipfs/
[8] https://discuss.ipfs.tech
[9] https://2022.ipfs.camp/
[10] https://2023.ipfs-thing.io/
[11] https://www.youtube.com/@IPFSbot
[12] https://www.veilid.com

Solution	Reference(s)	Privacy Goal	Available
Tor	(IPFS, 2020; IPFS Primer, 2020a,b)	Anonymity	✔
I2P	(RTrade Technologies Ltd, 2019; Mike Chu, 2023)	Anonymity	✔
Content encryption	(IPFS Docs, 2023e)	Content confidentiality	✔
DHTPIR	(Mazmudar et al., 2021)	Plausible deniability	
Double Hashing/ Reader Privacy	(Guillaume Michel, 2022)	Plausible deniability	
Bloom-Swap	(Daniel and Tschorsch, 2023)	Content anonymity	
Content Erasure	(Politou et al., 2020)	Right to erasure	

Table 1. Solutions for reader privacy, their respective references, the primary privacy goal covered, and whether they are available, i.e., ready to use as of today.

To summarize and answer **RQ1**: The anonymity of users and content providers is not preserved, which poses a serious privacy violation. Moreover, a standard implementation of the protocol allows the tracking of interactions, including the tracking of requested and stored content.

Reader Privacy

In this section, we answer **RQ2a**: *Which solutions exist to address privacy issues for users?* This problem is often referred to as reader privacy, as opposed to writer/provider privacy, which is addressed in the succeeding section. In addition to disclosing privacy risks inherent in IPFS (see previous section), Balduf et al. (2022) also proposed some high-level mitigation tactics:

1. Changing the PeerID more frequently
2. Anonymizing the node's IP
3. Reducing the amount of allowed connections
4. Limiting the lookup space to trusted nodes
5. Using salted hashes of CIDs to prevent the linking of the CID to the respective content if the CID is unknown
6. Adding noise to CID requests to provide plausible deniability

None of these techniques has been discussed in detail; however, many of them have been picked up by other researchers.

Currently, researchers are looking into ways to incorporate Tor, Nym[13], Freenet[14], and Signal[15] into the IPFS ecosystem in order to facilitate privacy-preserving communication (Protocol Labs, 2023). However, none of these projects are finished or have published works in progress. The list of currently supported projects can be found online[16] and mainly focus on PIR. Methods and projects presented in this section are categorized by their primary targeted privacy goal (see Tab. 1).

Anonymity

Tor with IPFS One of the most widely used PETs is the Tor network (Dingledine et al., 2004). It has been developed to preserve an individual's privacy while browsing the web. If that individual's client accesses a website via Tor, it is not possible to link that client to their actual IP address, thus preserving anonymity. This is achieved by routing the HTTP request through the Tor network before forwarding it to the actual server. The privacy provided by Tor is strong, and even the NSA has to admit that it "will never be able to de-anonymize all Tor users all the time" (David Murphy, 2013). Recent developments also lowered Tor's latency, although, for design reasons alone, it still lags behind conventional web browsing. The simplest way to use Tor in combination with IPFS is to access an HTTP gateway like `ipfs.io/ipfs/[CID]` with the Tor browser and replace [CID] with the actually desired CID. However, users can only download content this way.

[13] https://nymtech.net/
[14] https://www.hyphanet.org/index.html
[15] https://signal.org/
[16] https://grants.protocol.ai/blog/2023/private-retrieval-grant-2023-roundup/

A talk at the IPFS þing 2022 highlights some problems with incorporating Tor-like anonymization services into IPFS[17]:

1. Circuit selection, as nodes must not collude
2. Network structure, as the current P2P architecture would not provide dedicated entry and exit nodes
3. Changes to the IPFS software would be severe
4. Latency would be an issue due to the limited amount of users willing to participate, which could be alleviated with incentive structures

The official documentation suggests using a certain IPFS gateway on the Tor network (IPFS Primer, 2020a); however, this information seems to be deprecated; as of December 2023, the provided gateway does not work. The provided alternative is to run IPFS over Tor transport, which allows nodes to communicate over the Tor network, thus providing node anonymity (IPFS Primer, 2020b). However, this concept was stuck in the experimental phase and did not materialize (IPFS, 2020). Referring to the current list of projects working on PIR, none of them mention Tor, which suggests that accessing HTTP gateways via Tor is the only viable solution to combine Tor with IPFS.

I2P with IPFS Likewise to Tor, the Invisible Internet Project (I2P) has been introduced in order to preserve the users' anonymity while browsing the internet at a reasonable latency (Schimmer, 2009). Compared to Tor, the user base is much smaller, and the majority of it is in Europe (Mike Chu, 2023).

Similar to using a gateway to access IPFS via Tor, it is equally possible to access IPFS via I2P. Once installed, I2P can be used just like Tor to access IPFS HTTP gateways. One company that offers IPFS access over I2P is RTrade, which runs Temporal[18]—an open-source data storage API—and maintained an I2P gateway project (RTrade Technologies Ltd, 2019) (the last commit was in 2018). Using Temporal to access an HTTP gateway via I2P works by installing I2P and then accessing their gateway through I2P. This way, the user's anonymity is preserved, and access, but not file upload within IPFS, is possible.

> Achieving anonymity for all IPFS network interactions is currently not possible. The only way users have to not link their IP addresses to their data requests, is to access the IPFS network from outside, i.e, via gateways, and hide their IP using either Tor or I2P.

Plausible Deniability

Double Hashing/Reader Privacy The name *double hashing* stems from the fact that content is at the moment hashed twice already. Once to get the CID and the second time to get the bit-string representation that is used for the lookup. Since the bit-string representation directly links to the CID, thereby revealing the content a node is interested in, the idea is to find a bit-representation of the CID that does not directly link to the original CID, thus hiding the content requested from prying eyes. This idea has been presented on multiple occasions (IPFS Camp 2022[19], IPFS þing 2022[20]) and is specified by Guillaume Michel (2022). It is worth noting that the name has since been changed to *DHT reader privacy*.

The goal of the presented approach is to achieve k-anonymity and plausible deniability. k-anonymity means that the quasi-identifiers of an anonymized record within a database are indistinguishable from $k-1$ other records within the same database (Sweeney, 2002). In the context of IPFS, however, this translates to hiding the actual CID a peer is trying to fetch within $k-1$ other CIDs, thus providing plausible deniability. Referring to Fig. 2, if we assume the bit-string `001` points to a certain CID, a node would request all CIDs with the prefix `00`. The DHT server would now send the peers who have both bit-strings `000` and `001`, which would yield in 2-anonymity. In practice, these bit-strings would be much larger. Depending on where the prefix of the bit-string is cut off, this level of privacy is provided at the expense of network overhead. This overhead arises because instead of only providing the pointer to the specific content, all pointers beginning with the respective prefix are sent over the network.

While providing plausible deniability to the reader, it jeopardizes writer privacy. To recall, the provider record's key is the CID, which means that before *double hashing/reader privacy*, a node can only see

[17] https://www.youtube.com/watch?v=f85U8b5g-Ks
[18] https://temporal.cloud
[19] https://www.youtube.com/watch?v=VBlx-VvIZqU
[20] https://www.youtube.com/watch?v=ZPIDU1-JnVc

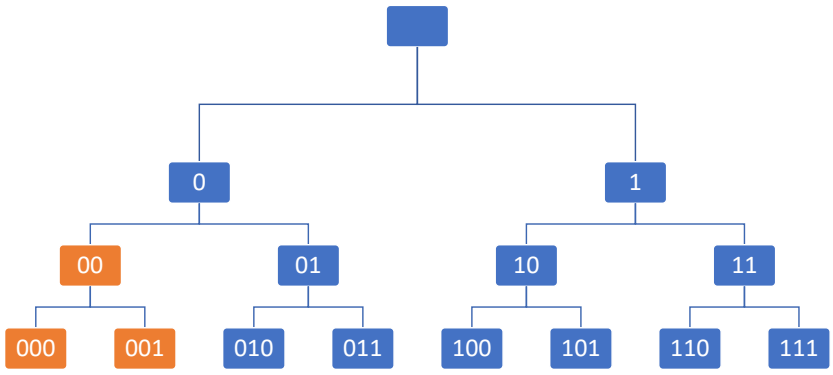

Figure 2. Example of *double hashing/reader privacy* providing $k = 2$-anonymity

PeerIDs that store known CIDs. This changes if providers send their pointers to the requesting node, as they include CIDs that were not previously known, which exposes more PeerIDs and violates provider privacy for the benefit of reader privacy.

To mitigate that, the proposed protocol includes symmetric and asymmetric encryption to preserve privacy. Content providers encrypt their PeerID using an encryption algorithm that includes the respective CID, exposing only the PeerID to nodes that know the CID. To further ensure that provider records cannot be faked, the encrypted provider record is signed so that the DHT server can verify their origin. Double Hashing/Reader Privacy is already defined in spec IPIP-373 (Guillaume Michel, 2022) but, as of today, is not yet integrated into IPFS.

DHTPIR Miti Mazmudar's talk[21] outlines the concept of DHTPIR (Mazmudar et al., 2021). The privacy goal of this protocol is to hide content-retrieval queries from intermediate nodes in order to encourage more clients to use IPFS for sensitive content in countries in which this content is censored. The main goal is to circumvent malicious nodes that want to censor content, differentiating the approach from *double hashing/reader privacy*. Consequently, the underlying threat model is twofold: (1) nodes that (are coerced to) log access to certain content and forward it to censors and (2) malicious nodes that either drop requests (eclipse attack (Prünster et al., 2022)) or forward them to the wrong node. The DHTPIR protocol employs the IT-PIR scheme, which requires at least two nodes that are not malicious. In order to fulfill this assumption, DHT nodes are grouped together in so-called quorums, such that most nodes in each group are benign (Sen and Freedman, 2012). Fig. 3 illustrates this. Several nodes are grouped together in quorums Q_1, Q_2, and Q_3. The assumed data flow is that node p wants to fetch content from Q_3, in which all nodes have a common DHT from which the content is fetched, i.e., sending seemingly random vectors to each node within the quorum that XORed together, allows p to retrieve the desired content. Referencing Fig. 3, if node p were to directly query the malicious node r, it could forward p's request to another malicious node, thereby tampering with the result or simply dropping the request and censoring the desired content. By using quorums, p does not only ask r but all nodes within Q_2.

A concept called threshold signatures is applied, such that a request reaching the quorum is only forwarded if a certain threshold of benign nodes' signatures is reached (Young et al., 2010). Note that all nodes within a quorum hold the same files.

The privacy-preserving file retrieval algorithm is then carried out using Perfect Hash Function (PHF) (Botelho et al., 2013), which is used to map document keywords to their respective indices. In the context of server nodes in IPFS the PHF is used to compute the content's index from its CID. The algorithm works as follows:

1. Identify the index of the file in the target quorum's database using PHF
2. If there are s number of servers within the quorum, compute s vectors which combined result in the desired content
3. Send one vector to each node within the quorum and obtain the PIR responses

[21] https://www.youtube.com/watch?v=1cd4t9OL0iM

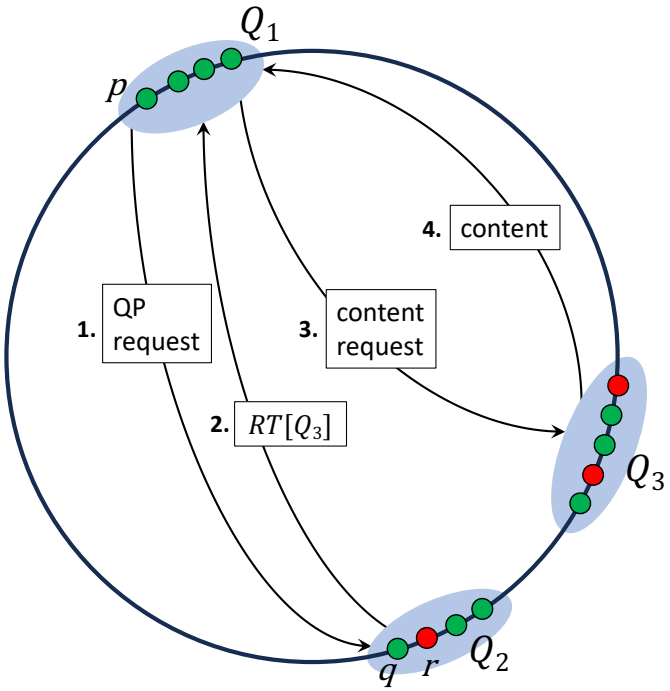

Figure 3. Simplified version of the QP protocol (Mazmudar et al., 2021). Quorums are illustrated by blue ovals, benign nodes are green, and malicious nodes are red. Node p wants to fetch content that is stored in Q_3 with no direct connection. 1. Q_1 requests Q_2's routing table including Q_3. 2. Q_2 returns Q_3's routing entry as the malicious node r is in the minority and can not "convince" the others to drop the request. 3. Node p queries Q_3 via IT-PIR for the desired content that 4. is sent to p.

4. Reconstruct the file using these responses

Submission for integration of DHTPIR into IPFS has been proposed but has not yet been carried out.

> Plausible deniability is the most straightforward privacy goal that can be achieved within IPFS. The main idea behind both presented methods is to obscure the actual data request by adding "noise", i.e., requests of undesired data blocks. The main differences between the methods are the trust model and the way the noise is distributed.

Content Privacy

Bloom-Swap Bloom-Swap is a method for privacy-enhanced content discovery similar to *double hashing/reader privacy* (Daniel and Tschorsch, 2023). It has the same goal: Find a block within the network without broadcasting a peer's interest to the whole network. Their approach relies on replacing the request of a specific CID, with a noisy request containing the desired CID mixed with all other CIDs of a provider. In order to reduce the network overhead and to provide plausible deniability to the server, the list is hashed onto a Bloom filter (Bloom, 1970). Instead of querying the desired block directly, a node requests a Bloom filter of all available blocks on the server—hence, the name. If the Bloom filter suggests that the desired block is available, the client can now request this block directly. Bloom filters are probabilistic in nature, and false positives are possible. In this case, the server will respond that the block is not available and the node's privacy is not maintained. However, false negatives are ruled out: If the Bloom filter returns a negative for the desired block, it is certain that it is not present, and nothing will be requested. Note that this reveals the amount of blocks stored on the server—information that is private in traditional Bitswap. To the best of our knowledge, Bloom-Swap is purely theoretical at the moment, and there are no integration plans.

Bloom-Swap is an efficient way of obscuring data requests from intermediate nodes. Using Bloom filters substantially reduces network overhead. However, it does not protect the requesting node's privacy from the node that actually provides the content.

Content Erasure

Delegated Content Erasure The previously presented approaches cover the protection goals of *anonymity* and *plausible deniability*. One important privacy objective that can also be found in the General Data Protection Regulation (GDPR) is the right to be forgotten (GDPR, Article 17). Due to the decentralized structure of IPFS, it is currently almost impossible for any data controller to satisfy this right. This issue has also been discussed on the IPFS subreddit[22], and answers suggest not to upload anything you deem private, as there is no way to ensure permanent erasure. As stated earlier, the garbage collector deletes any unpinned content from a node's cache. However, this does not mean that the whole network is traversed and copies of that file are also deleted. For example, user A shares a holiday picture via IPFS and pins it. A's peers download it and pin it themselves. If A unpins the photo and does not want it to be available via IPFS, A would have to tell his peers to unpin it, who in turn have to tell everyone they shared it with. A file is only *forgotten* and no longer available on IPFS if no node pins it and all cached copies are removed by the garbage collector (Politou et al., 2020).

The proposed protocol now tackles this issue by including several cryptographic keys to ensure that (1) content is deleted from all nodes that currently possess it and (2) only authorized users, i.e., content owners, are allowed to delete the respective content. The protocol has the following steps:

1. Each user generates a secret symmetric master key mk.
2. If c is published, the user calculates the content-based key k by encrypting the hash of c using mk: $k = E_{mk}(h(c))$.
3. The owner calculates the proof of ownership s by encrypting the hash of c using k: $s = E_k(h(c))$.
4. The user now publishes the tupel (c,s), i.e., the content and its respective proof of ownership.
5. If the user wants to permanently delete c, they send a deletion request using the tupel $(h(c),k)$.
6. A node can verify the ownership by calculating $E_k(h(c))$ themselves and verifying with the already published proof of ownership s.
7. Upon successful verification, the content is removed, and the deletion request is forwarded.

Note that no one except for the actual content owner can issue a deletion request if mk is kept secret. Moreover, publishing k during the deletion request in step 5 should not compromise mk, i.e., an attacker that knows (the now public) k should not be able to infer mk from it. This protocol was suggested in 2020 but has not yet been implemented in IPFS.

Permanent content erasure is a problem that enjoys less attention than other privacy-related topics; however, it is an important issue to consider in a decentralized network. The proposed protocol is a well-structured way to alleviate this issue, but its changes to the IPFS network would presumably be too severe.

Research Question RQ2a

After having gone through the possible solutions to achieve user privacy within the IPFS network, we can now answer **RQ2a**: *Which solutions exist to address privacy issues for users?* The only viable solution today is accessing the IPFS network anonymously from the outside, i.e., via an HTTP gateway using Tor or I2P. Due to the larger user base, Tor is the most usable solution. The apparent downside is the fact that users are not participating in the network if they just access it via HTTP, which means that these solutions only offer anonymity for content readers. The most promising solutions target the privacy goal of *plausible deniability*. This is likely because adding noise via superfluous content requests is comparatively easy, and the only difference between the methods is the way how this noise is generated.

[22] https://www.reddit.com/r/ipfs/comments/9puvpo/

Provider Privacy

In this section, we address **RQ2b**: *Which solutions exist to address privacy issues for content providers?* To achieve privacy for content providers, also called writers, their anonymity has to be preserved. Outside of IPFS, this can be achieved using Onion services, also known as hidden services or the "Darknet". Using Onion services means that a server can provide content without risking exposure of the IP address.

In IPFS, provider privacy is a more complicated problem than user privacy. Obscuring data requests via gateways and anonymity networks or providing plausible deniability by flooding content requests is impossible for content providers. Moreover, projects addressing user or reader privacy explicitly put provider or writer privacy out of scope (Yiannis Psaras, 2023).

Methods that provide plausible deniability to users can also be argued to improve provider privacy[23]. For example, a node operator storing sensitive content might be compelled to log content requests and leak that information. However, due to the added noise or carefully constructed query vectors, node operators can no longer tell which content a user actually wants. This provides plausible deniability to the providers themselves and can incentivize more nodes to provide sensitive content.

To answer **RQ2b**, no methods or projects exist at the moment that address provider privacy apart from being a byproduct of user privacy. Concretely, it is currently not possible to anonymize the content provider's IP address but limited plausible deniability can be achieved.

DISCUSSION

IPFS is not built with privacy in mind. This becomes apparent by how content is published and accessed, exposing both the user's IP and desired content. A malicious actor is able to breach reader privacy and provider privacy by tracking content requests, allowing for surveillance and censorship.

Reader Privacy

Reader privacy methods focus on (1) anonymity, (2) plausible deniability, (3) content anonymity, and (4) content deletion.

Efforts to incorporate peer anonymity hide the user's IP address by accessing IPFS gateways via Tor or I2P. This does not address the privacy issues directly, as users are not part of the IPFS network when using a gateway; however, it is the only option for now to fetch data anonymously via IPFS. Projects that seek to integrate Tor into the IPFS network are still in progress or seemingly abandoned in favor of other anonymity techniques, which are not yet publicly available.

Methods that provide plausible deniability to users do this by adding "noise" to the data request. By not only requesting the CID blocks the user actually desires but also adding blocks of other CIDs to the request, a user can plausibly deny that a specific CID has been requested. The network overhead depends on the individual privacy needs and is directly proportional. The more blocks a client requests, the less likely it is to infer the actually desired content. This technique (or any other technique that relies on requesting more blocks than needed) can not guarantee a diversity of requests. Hence, if out of ten blocks, only one is popular and nine are never requested, it is still possible for an attacker to make an educated guess about the actually desired content within the noise. Nevertheless, the privacy increase of these methods is substantial with regard to the comparably low network overhead and limited changes to the IPFS protocols.

Content anonymity describes the hiding of desired content from intermediate nodes. The presented method obscures the peer's content request by fetching a list of CIDs and checking if the server has the desired content. A Bloom filter where false positives are possible is used to reduce traffic overhead and to provide probabilistic privacy that becomes stronger the larger the filter is. The peer's privacy is only protected from intermediaries, and thus, the content provider knows of the peer's request. A combination with a plausible deniability method could alleviate this problem.

Finally, the protocol presented to erase content does not deal with the immediate threat to a user's privacy but still covers an important missing feature of IPFS: the permanent erasure of content. Permanently deleting files is a crucial piece in preserving an individual's privacy, as it is important to account for changes in privacy attitude or perceived sensitivity of the published content. Currently, there is no way to permanently delete content; however, the proposed method could solve this issue elegantly. Incorporating proof of ownership deters misuse of this deletion method. The changes to the protocol are, however, more severe, making it questionable whether permanent content erasure will be implemented in IPFS.

[23] https://www.youtube.com/watch?v=1cd4t9OL0iM

Provider Privacy

The lack of research into methods regarding provider privacy is not surprising, given the focus on users' private information retrieval by Protocol Labs. Also, it is difficult to argue for provider privacy given the implications Onion services had on Tor's reputation as a network for illegal operations. There is limited plausible deniability for providers as a byproduct of efforts to increase user privacy, but these effects are limited. Should provider privacy come into focus, it is still an open question how to achieve it. Current projects regarding Mixnets could improve provider privacy, but at the moment, there is no public information available about them.

CONCLUSION

Privacy is an often overlooked concept in many areas, and IPFS is no exception. Due to the protocol's design, it seems to be impossible to grant anonymity to all nodes within the network. The solutions presented in this paper are either ways to circumvent these issues, e.g., by using Tor or I2P from outside the network, or band-aids, which add noise in order to obfuscate data requests which results in plausible deniability. Most of the presented solutions are only theoretical so far, and only an HTTP gateway via Tor or I2P can be used today to load content anonymously from the IPFS network. One solution to providing plausible deniability—*double hashing/reader privacy*—is specified but not yet integrated. The need for additional privacy measures becomes apparent when looking at Protocol Lab's funding of $750,000 for solutions related to private information retrieval[24]. These projects target reader privacy only, so provider privacy beyond limited plausible deniability remains an open issue in the near future.

REFERENCES

Angel, S., Chen, H., Laine, K., and Setty, S. (2018). PIR with Compressed Queries and Amortized Query Processing. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 962–979. IEEE Computer Society.

Balduf, L., Henningsen, S., Florian, M., Rust, S., and Scheuermann, B. (2022). Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS. In *Proceedings of the 42nd International Conference on Distributed Computing Systems (ICDCS)*.

Benet, J. (2014). IPFS—Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*.

Bloom, B. H. (1970). Space/Time Trade-Offs in Hash Coding With Allowable Errors. *Communications of the ACM*.

Botelho, F. C., Pagh, R., and Ziviani, N. (2013). Practical Perfect Hashing in Nearly Optimal Space. *Information Systems*.

Carson Farmer (2018). Swapping bits and distributing hashes on the decentralized web.

Chaabane, A., De Cristofaro, E., Kaafar, M. A., and Uzun, E. (2013). Privacy in Content-oriented Networking: Threats and Countermeasures. *ACM SIGCOMM Computer Communication Review*.

Daniel, E. and Tschorsch, F. (2023). Privacy-Enhanced Content Discovery for Bitswap. In *Proceedings of the IFIP Networking Conference (IFIP Networking)*.

Daniel Norman (2022). A Practical Explainer for IPFS Gateways - Part 2.

David Murphy (2013). How the NSA Takes On the Tor Project.

Dingledine, R., Mathewson, N., Syverson, P. F., et al. (2004). Tor: The Second-Generation Onion Router. In *Proceedings of the USENIX Security Symposium*.

Dirk McCormick and Edgar Lee (2020). New improvements to IPFS Bitswap for faster container image distribution.

Filecoin Docs (2023). Filecoin and IPFS.

Gan, M. F., Chua, H. N., and Wong, S. F. (2019). Privacy Enhancing Technologies Implementation: An Investigation of its Impact on Work Processes and Employee Perception. *Telematics and Informatics*.

Guillaume Michel (2022). IPIP-373: Double Hash DHT Spec #373 .

Henningsen, S., Florian, M., Rust, S., and Scheuermann, B. (2020a). Mapping the Interplanetary Filesystem. In *IFIP Networking Conference (Networking)*.

Henningsen, S., Rust, S., Florian, M., and Scheuermann, B. (2020b). Crawling the IPFS Network. In *IFIP Networking Conference (Networking)*.

IPFS (2020). Tor onion integration.

[24] https://grants.protocol.ai/blog/2023/private-retrieval-grant-2023-roundup/

IPFS Docs (2022). Work with blocks.
IPFS Docs (2023a). Bitswap.
IPFS Docs (2023b). Frequently Asked Questions—IPFS and Filecoin.
IPFS Docs (2023c). IPFS Gateway.
IPFS Docs (2023d). Merkle Directed Acyclic Graphs (DAGs).
IPFS Docs (2023e). Privacy and Encryption.
IPFS Primer (2020a). Lesson: Access IPFS content through Tor gateways (experimental).
IPFS Primer (2020b). Lesson: Run IPFS over Tor transport (experimental).
Kaaniche, N., Laurent, M., and Belguith, S. (2020). Privacy Enhancing Technologies for Solving the Privacy-Personalization Paradox: Taxonomy and Survey. *Journal of Network and Computer Applications*.
Kushilevitz, E. and Ostrovsky, R. (1997). Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, page 364. IEEE Computer Society.
Matt Ober (2019). IPFS Privacy.
Mazmudar, M., Gurtler, S., and Goldberg, I. (2021). Do You Feel a Chill? Using PIR Against Chilling Effects for Censorship-Resistant Publishing. In *Proceedings of the 20th Workshop on Privacy in the Electronic Society*.
Mike Chu (2023). Tor vs I2P: How Do They Compare?
Nizamuddin, N., Salah, K., Azad, M. A., Arshad, J., and Rehman, M. (2019). Decentralized Document Version Control Using Ethereum Blockchain and IPFS. *Computers & Electrical Engineering*.
Politou, E., Alepis, E., Patsakis, C., Casino, F., and Alazab, M. (2020). Delegated Content Erasure in IPFS. *Future Generation Computer Systems*.
Protocol Labs (2023). RFP-014: Private retrieval of data.
Prünster, B., Marsalek, A., and Zefferer, T. (2022). Total Eclipse of the Heart–Disrupting the {InterPlanetary} File System. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*.
RTrade Technologies Ltd (2019). A plugin for presenting an IPFS gateway over i2p.
Samonas, S. and Coss, D. (2014). The cia strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*.
Schimmer, L. (2009). Peer Profiling and Selection in the I2P Anonymous Network. In *Proceedings of the Privacy Enhancing Techniques Convention (PetCon)*.
Sen, S. and Freedman, M. J. (2012). Commensal Cuckoo: Secure Group Partitioning for Large-Scale Services. *ACM SIGOPS Operating Systems Review*.
Shen, Y. and Pearson, S. (2011). Privacy Enhancing Technologies: A Review. *Hewlet Packard Development Company*.
Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*.
Tenorio-Fornés, A., Jacynycz, V., Llop-Vila, D., Sánchez-Ruiz, A., and Hassan, S. (2019). Towards a Decentralized Process for Scientific Publication and Peer Review Using Blockchain and IPFS. *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*.
Tourani, R., Misra, S., Mick, T., and Panwar, G. (2017). Security, Privacy, and Access Control in Information-centric Networking: A Survey. *IEEE Communications Surveys & Tutorials*.
Westin, A. F. (1967). *Privacy and Freedom*. Atheneum.
Xu, Q., Song, Z., Goh, R. S. M., and Li, Y. (2018). Building an Ethereum and IPFS-Based Decentralized Social Network System. In *Proceedings of the 24th International Conference on Parallel and Distributed Systems (ICPADS)*.
Yiannis Psaras (2023). The IPFS DHT Reader Privacy Upgrade.
Young, M., Kate, A., Goldberg, I., and Karsten, M. (2010). Practical Robust Communication in DHTs Tolerating a Byzantine Adversary. In *Proceedings of the 30th International Conference on Distributed Computing Systems*.