Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Privacy-Preserving Content Discovery for Bitswap Using Random Walks

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

eingereicht von: Manuel Wedler
geboren am:
geboren in:

Gutachter/innen:

eingereicht am: ..... verteidigt am: .....

Abstract

The peer-to-peer file storage system IPFS relies on the Bitswap protocol for the exchange of data. Bitswap reveals both the requested content's identifier and the requestor's identity to neighbors during content discovery, posing privacy risks by potentially enabling the tracking of user interests. This thesis addresses these privacy concerns by proposing a modification to Bitswap, enabling source obfuscation through random-walk-based forwarding. The forwarding introduces plausible deniability into the protocol, preventing adversaries from identifying users' interests. Through an experimental evaluation, the protocol modification demonstrates enhanced privacy for Bitswap users while maintaining acceptable performance levels. However, the protocol exhibits a vulnerability under the assumption that adversaries can craft arbitrary messages, as the disclosure of the request's originator can be triggered.

Contents

1. Introduction

Since earlier systems like Napster and BitTorrent were introduced over two decades ago, the trend of newly emerging peer-to-peer (P2P) data networks continues [10]. The InterPlanetary File System (IPFS) [3] represents a modern approach to decentralized file storage systems. Through its utilization of a P2P network, anyone can participate by storing data within the system or providing storage capacity. IPFS incorporates a distributed hash table (DHT) where data is stored across multiple nodes and addressed by its content. With these building blocks, IPFS strives to enhance the web's resilience, support content's resistance to censorship, and expedite file transfers [51]. While these properties are desirable for a decentralized file storage system, a recent study by Balduf *et al.* [2] proved that IPFS lacks another desirable attribute. Their study demonstrated how IPFS users' privacy could be compromised by outlining a methodology for monitoring data requests within the IPFS network. The demonstrated attacks include identifying users interested in a specific data item, tracing the data items requested by a user, and checking whether a user had previously requested a particular data item. This study highlights that the users' privacy is vulnerable and a passive observer could identify the interest of users.

The source of IPFS's privacy problems lies in Bitswap [42], the protocol responsible for content exchange within IPFS. While Bitswap's most prominent use is in IPFS, it is also integrated into the block synchronization protocol of Filecoin [53]. Bitswap operates on data items, each of which is assigned a content identifier (CID) by applying a hash function to the data itself. This CID serves as the means to address the data item within the P2P network. Besides data exchange, the Bitswap protocol enables content discovery. Bitswap queries all neighbors for a CID to find providers for the desired content. Only if this initial request yields no providers, Bitswap asks the DHT of IPFS for content providers. While sending the request to all neighbors enhances the DHT lookup in terms of fault tolerance and speed [23, 63], it also represents a privacy risk. In this initial query, the requestor reveals its interest in a CID to all neighbors, enabling profiling by curious or malicious nodes. Furthermore, querying the DHT involves additional requests to nodes, which in turn can learn about a node's interests, thereby presenting another opportunity for monitoring users. The goal of this thesis is to investigate how Bitswap can be modified to enhance privacy.

IPFS and, thus, Bitswap do not implement any privacy mechanisms by design [50]. Hence, multiple approaches to enhance the privacy of Bitswap have been studied recently [9, 12]. While all of these solutions improve privacy, they also encompass a trade-off. The integration of trickling [9] produces an increased load on the network. The approaches in [12], which incorporate private set intersection, add computational overhead to the protocol. The Bloom filter approach of [12] reduces provider privacy. Therefore, further options for integrating privacy-preserving mechanisms into the Bitswap protocol should be evaluated. The root of Bitswap's privacy issue is that a requestor reveals its interest in a CID through the request and an adversary can be certain that the sender of the request is also its originator. A technique that offers an effective means of obfuscating the source of a message in a P2P network is the

utilization of random walks. Their usage for delegating a user's request to a proxy in a distributed system has been studied for decades [56]. Recently, the Dandelion [5] and Dandelion++ [18] protocols incorporated random walks to introduce privacy-enhancing transaction propagation into Bitcoin [36]. In the Dandelion and Dandelion++ protocols, transactions are relayed through the P2P network over a random walk to a proxy, which eventually diffuses the transaction. Any node can plausibly deny that it is the originator of the transaction, as it could also be relaying the transaction. Thus, the source of the transaction is obfuscated.

In this thesis, the approach for transaction propagation of Dandelion and Dandelion++ is applied to the Bitswap protocol. The resultant modification of Bitswap, referred to as *RaWa-Bitswap*, utilizes random walks for preserving users' privacy. In RaWa-Bitswap, requests are forwarded on a random walk to a proxy that acquires content providers. Information about these providers is then routed back to the requestor which can directly download the data from a content provider. By forwarding requests, the nodes querying for providers gain plausible deniability, preventing adversaries from identifying the source of the request. The interest in a CID is ideally revealed to only one peer, the content provider that transfers the data to the requestor. For the purpose of increased mixing of messages, the protocol proposes to forward messages in an approximated regular subgraph constructed from the network graph.

RaWa-Bitswap is evaluated in a simulation-based setup and compared to Baseline Bitswap in terms of their privacy and performance properties. Despite not achieving ideal characteristics, RaWa-Bitswap significantly enhances privacy for Bitswap users while preserving performance under the tested conditions. It notably outperforms Baseline Bitswap, showcasing reduced detection probabilities of individual nodes' interests and an increased level of plausible deniability. Forwarding messages within the complete network graph, rather than utilizing a subgraph, yields the best privacy results under a strong adversarial model. Furthermore, the experimental setup proves that RaWa-Bitswap exposes a vulnerability when assuming an adversary capable of crafting arbitrary messages. This vulnerability stems from the disclosure of a request's originator through `WANT-BLOCK` messages and the absence of interest obfuscation. The simulations indicate that this vulnerability enables a high probability of detecting the interest of a peer, while the level of plausible deniability remains relatively high. The main contributions of this work are the design of a random-walk-based forwarding protocol that enhances privacy under a request-response model, the modification of Bitswap into this protocol, and an experimental setup that demonstrates the protocol's privacy enhancement, but also its limitations under certain assumptions.

The remainder of the thesis is structured as follows: Section 2 explores the Bitswap protocol and the root of its privacy issues. The fundamental algorithms utilized by the proposed protocol modification to enhance privacy are outlined in Section 3. The protocol modification itself is specified, and its design is reasoned in Section 4. The experimental evaluation of the protocol's properties is presented in Section 5. Section 6 provides an overview of related work. Section 7 concludes the thesis.

2. Bitswap

Bitswap is a message-based protocol which is used to exchange content-addressable data in a P2P network [14]. In this section, the Bitswap protocol is described in-depth. Furthermore, the privacy issue of Bitswap, which should be solved in this work, is defined by giving a problem statement at the end of this section.

2.1. The Bitswap Protocol

Bitswap pursues two goals: discovering which nodes have content and actually fetching the content [63]. The content is split into chunks, called *blocks*, which can be linked to each other. Discovery and exchange of these blocks is handled by communicating with other peers through messages. The following describes the structure of the blocks, the specific messages and what pattern Bitswap follows in the message exchange for content retrieval. Bitswap's characterization in this section refers to the latest version of the protocol, v1.2.0 [42].

2.1.1. Content Blocks

Bitswap operates on *blocks* which carry the content that should be exchanged via the protocol. These blocks are addressed by an immutable identifier, called *content identifier* (CID), which is constructed from the content itself [46]. A CID consists of the codec information and a *multihash*. A multihash is the hash of the content included in the block, the digest length and used hash function [48]. This makes the CID immutable and unique for each block.

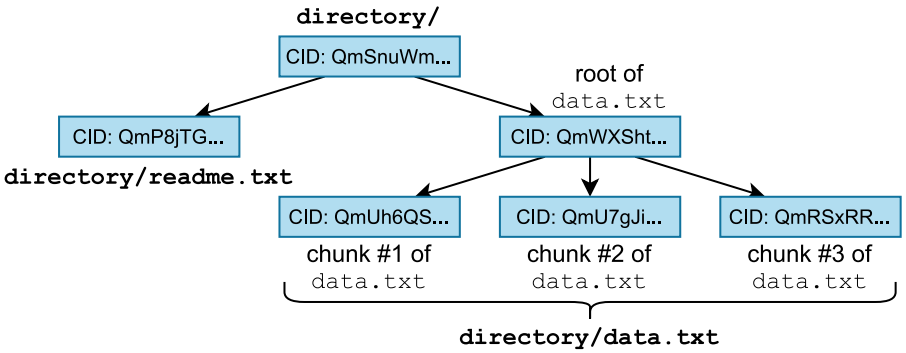

Figure 1: An example of a directory represented as a Merkle DAG.

Blocks can be linked to each other by constructing a *Merkle Directed Acyclic Graph* (DAG) [49, 52]. Any block consists of data and links to other blocks by incorporating their CIDs. In contrast to a Merkle tree, all nodes in a Merkle DAG can carry a payload. Merkle DAGs are constructed from the leaves, because the CIDs of the children must be known to construct the parent node. The immutability property of the CID makes a whole Merkle DAG immutable. For example, one file can be split into multiple blocks

forming a Merkle DAG, or a file directory can be represented as a Merkle DAG. An example of how a directory containing two files would be represented as a Merkle DAG can be seen in Figure 1. By representing content as a Merkle DAG, the content can be identified by the CID of the DAG root. This root CID needs to be requested with Bitswap to retrieve the content.

2.1.2. Messages

The Bitswap protocol operates by exchanging messages between peers. The Bitswap specification [42] distinguishes between client and server messages. The client messages are used to request content from peers, while server messages are used to respond to these requests. Multiple messages can be wrapped in the same message envelope. This allows sending server and client messages in one envelope at the same time, as one Bitswap node can take both roles. A message envelope can include one *wantlist*, multiple *block presences*, and multiple blocks. The wantlist can contain multiple entries of the client messages `WANT-HAVE`, `WANT-BLOCK`, and `CANCEL`. These messages signal if a requestor wants to obtain the block corresponding to a specified CID. Block presences are the server messages `HAVE` and `DONT-HAVE`, which are used to indicate if a block is stored by the sender of the message. Blocks are sent by the server message `BLOCK`. All of the messages are explained more precisely in the following [14]:

- Client messages:
  - `WANT-HAVE`: Asks if a peer has the block corresponding to a specified CID. The receiver announces if it stores the block or not by responding with a `HAVE` or a `DONT-HAVE` message.
  - `WANT-BLOCK`: Requests to transfer the block corresponding to a specified CID. If the receiver stores the block, it sends the block back in a `BLOCK` message. The receiver can report that it does not store the requested block by responding with a `DONT-HAVE` message.
  - `CANCEL`: Cancels the previous want for a specified CID. For this CID, the receiver should not send a response anymore.
- Server messages:
  - `HAVE`: Announces that the block corresponding to a specified CID is stored locally.
  - `DONT-HAVE`: Notifies that the block corresponding to a specified CID is not stored locally.
  - `BLOCK`: Transfers a requested block by wrapping it in the message.

2.1.3. Interaction Pattern

In order to retrieve content, Bitswap follows a pattern when exchanging the aforementioned messages. The following paragraph describing Bitswap's interaction pattern is

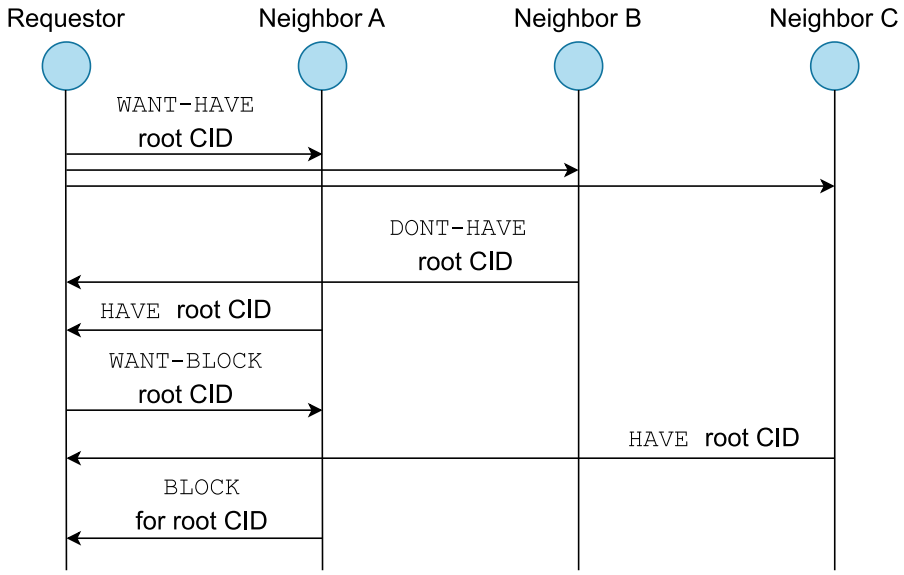

Figure 2: Bitswap's message exchange.

based on the work by De la Rocha *et al.* [14]. When a user requests content via Bitswap, a new *session* is initiated. A session keeps track of the peers that may possess the blocks for the requested file or directory, identified by the root CID. To discover these peers, a Bitswap node sends out a `WANT-HAVE` message for the root CID of the desired content to all its connected neighbors. Upon receiving the `WANT-HAVE` message, the receiving peers check if they store the block for the specified CID, and in this case respond with a `HAVE` message. Any peer responding with a `HAVE` message is added to the session by the requesting node. Once the requestor receives the first `HAVE` message, it sends a `WANT-BLOCK` message to the peer that sent the `HAVE` message. This `WANT-BLOCK` message requests the block corresponding to the root CID. The requested block is transferred to the requestor in a `BLOCK` message. The message exchange up to this point is illustrated in Figure 2. After receiving the first block, Bitswap starts to traverse the Merkle DAG structure in order to retrieve all the blocks. From the root CID block, the CIDs of the next level of the DAG are known. For each of these second-level CIDs, a `WANT-HAVE` and a `WANT-BLOCK` message are constructed. The `WANT-BLOCK` request is optimistically sent to one selected peer from the session. At the same time, Bitswap passes the `WANT-HAVE` to the rest of the session peers. If the optimistically chosen peer does not possess the block, one of the peers responding with a `HAVE` can be queried, as they certainly have the block. The probability of a peer being selected for a `WANT-BLOCK` is proportional to the number of blocks successfully transferred by that peer during the session. If the chosen peer responds with a `DONT-HAVE` message, another session peer is selected to send the `WANT-BLOCK` request. Once the requestor receives a block, it sends a `CANCEL` message to all peers previously requested for that block. In order to receive all blocks of the requested content, the entire Merkle DAG is iteratively traversed by discovering blocks through `WANT-HAVE` and downloading blocks by `WANT-BLOCK` messages. As long

as the session has not received a block for a desired CID, it resends any `WANT-HAVE` and `WANT-BLOCK` requests for the CID to all previous recipients every 30 seconds [39]. This is because peers may have acquired the block in the meantime. Peers that consistently respond with a `DONT-HAVE` message in multiple subsequent requests are pruned from the session. In scenarios where none of the connected peers possess a requested block, Bitswap utilizes a content routing subsystem, specifically a distributed hash table (DHT) in the case of IPFS. The content routing subsystem is queried for one CID to discover peers which have the desired content. The session is then updated with this new information, making future requests for the block feasible.

By analyzing the Bitswap reference implementation [40], several additional fallback mechanisms were identified for the session. The session employs the *idle tick* to query the content routing subsystem for one desired CID in case no blocks have been received for a certain period. The idle tick initiates this query when the session hasn't received a block within 1 second after session initialization or after an unusually long time since the last received block. To achieve this, the session tracks the latency for block retrieval and calculates an estimate of the typical block retrieval time. While Bitswap is in operation, new peers may connect, which potentially can also have the desired content. Therefore, the session periodically sends a `WANT-HAVE` request for a random desired CID to all newly connected peers. This functionality is controlled by the *periodic search timer*, which triggers every 60 seconds. Additionally, the content routing subsystem is queried for the random CID when this timer triggers. Generally, in situations where neighbors are unresponsive to client messages, the session assumes that a peer does not possess a block if no response is received within 5 seconds.

2.2. Problem Statement

Content discovery in the case of Bitswap means finding the *PeerID* of a peer that can provide the block for a CID. A PeerID is the multihash of the peer's public key and is used to identify the peer [45]. While Bitswap could also rely fully on the content routing subsystem for content discovery, it also performs its own content discovery by sending `WANT-HAVE` messages to all connected neighbors. This is done to expedite the retrieval of requested content [63]. As described in Section 2.1.3, a request that includes a CID the user is interested in is sent to all connected neighbors in the initial query of a session and every 30 seconds thereafter. Consequently, the user's interest in the content associated with the CID is revealed to all neighbors. IPFS nodes using the default settings of the reference implementation maintain 32 to 96 direct connections with other peers [41]. Curious or malicious nodes among the neighbors can exploit this information for profiling of the user. As mentioned in Section 1, Balduf *et al.* [2] demonstrated how an attacker could exploit the information leakage of Bitswap. They defined three attacks and conducted them on public IPFS gateways. These attacks facilitated the discovery of nodes that are interested in the content specified by a CID, the tracking of which CIDs a node requests, and the identification of whether a node has previously requested a CID. If Bitswap's own content discovery using the `WANT-HAVE` requests fails, the content routing subsystem is queried. In the case

of IPFS, the content routing subsystem is a variant of the Kademlia [31] DHT. To retrieve content providers from the Kademlia DHT, a lookup algorithm is employed, which repeatedly queries other peers for the hash of a CID. This further reveals the user's interest in the content to additional peers and represents another opportunity for monitoring users.

In summary, the privacy issue with Bitswap arises from the fact that a requestor discloses the CID it is interested in through the request and the receiver knows that the sender of the request is also its originator. The content discovery involves requests to many peers, all of whom learn the user's interest in a CID. The data exchange in Bitswap can be narrowed down to the direct interaction between two nodes, and therefore does not reveal the interest in a CID to as many peers as is the case with the content discovery. Hence, the goal of this thesis is to improve the privacy of the content discovery of Bitswap. Content discovery requests should not be linkable to their originators. This can be achieved by delegating the content discovery to a proxy selected through a random walk. The request can be forwarded a random number of hops, obfuscating the source of the request. Any request a peer sends could also be relayed from a different peer. As a result, all peers gain plausible deniability. An introduction to random walks in the context of P2P systems is given in the next section.

3. Fundamental Algorithms

Random walks can help provide privacy to the users of P2P systems. In this section, an introduction to random walks in the context of P2P networks is given by demonstrating their usage in the Dandelion protocol [5]. A requirement for Dandelion to provide anonymity guarantees is its operation on regular graphs. At the end of this section, the distributed construction of regular graphs in P2P networks is shown.

3.1. Random Walks on P2P Networks

Random walks can be characterized as stochastic processes, in which an entity moves through a sequence of steps within a defined space [59]. When applied to a graph, they formalize a random selection of connected nodes along a path of variable length within the graph [30]. In this process, starting from a particular node, a neighboring node is selected at random. This selection procedure is iteratively performed for each chosen node until the random walk eventually terminates at a random point. In the context of P2P networks, the concept of random walks has been adopted by various protocols to enhance users' privacy. A message can be forwarded through the network's graph on random hops to increase the difficulty for attackers to trace the originator of a message. When a message is forwarded with a random walk, it is not certain that a sender of a message is also the message's originator, because the sender could have forwarded it from another peer [56].

A notable example of a P2P protocol that utilizes random walks for enhancing users' privacy is Dandelion [5]. Dandelion introduces privacy-enhancing transaction propagation with provable anonymity guarantees into Bitcoin [36]. Instead of diffusing a transaction directly, the transaction is forwarded through the P2P network on a random walk. The last hop serves as a proxy and diffuses the transaction. This obfuscates the identity of the transaction's originator, making it difficult for adversaries to trace the transaction back to the originator.

The Dandelion protocol consists of two distinct phases: an anonymity phase and a spreading phase. In the anonymity phase, Dandelion makes use of a privacy-subgraph that is constructed from the P2P graph of Bitcoin. The privacy-subgraph is built in a distributed fashion, in a way that the topology is not publicly known in the network. For propagating a transaction, a node sends the transaction to one neighbor within the privacy-subgraph. The receiving neighbor then randomly decides with a given probability either to relay the transaction to a successor within the privacy-subgraph or transition into the spreading phase and act as a proxy. This process of forwarding the transaction is repeated until a peer decides to proceed to the spreading phase. In the spreading phase, the proxy diffuses the transaction to all its neighbors in Bitcoin's complete P2P graph. The two phases are illustrated in Figure 3.

Dandelion operates on a privacy-subgraph in the anonymity phase because transaction mixing would be less in the complete graph. Relatively more transactions are propagated over the same path if only a subset of the graph's edges can be used for forwarding. An assumption that the authors of Dandelion made for achieving its formal

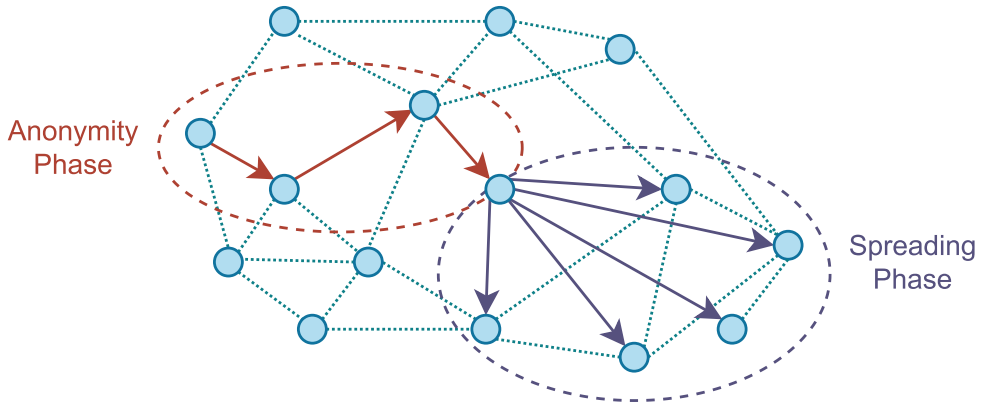

Figure 3: Transaction forwarding in the two phases of Dandelion.

anonymity guarantees is that the privacy-subgraph is unknown to adversaries. To prevent adversaries from learning the subgraph, it changes frequently. Every node runs the algorithm for picking new successors every 9 minutes. The construction does not need to be synchronized between the peers because a fully-distributed, asynchronous algorithm is used. The privacy-subgraph of Dandelion is a line graph (*i.e.*, a directed 2-regular graph) with one incoming and one outgoing edge per node. The authors of Dandelion later published an improvement to the protocol called Dandelion++ [18]. The main change is that a directed 4-regular graph functions as the privacy-subgraph, resulting in each node having two successors and two predecessors. Thus, when forwarding a transaction, a node randomly decides for one of the two successors or the diffusion. They chose to utilize a new privacy-subgraph in the protocol enhancement because they no longer assume that adversaries are unable to learn the privacy-subgraph. The 4-regular graph should optimize the formal anonymity guarantees between the two cases of a known and an unknown privacy-subgraph. The next section shows how regular graphs can be built in a distributed fashion.

3.2. Distributed Construction of Regular Graphs

To provide high transaction mixing in Dandelion and Dandelion++, transactions are propagated on a random walk through directed regular graphs. The construction of these privacy-graphs should be distributed and asynchronous in order to prevent adversaries from easily learning the graphs. The authors of Dandelion evaluated a few options for constructing exact line graphs. For example, [27] creates a 2-regular graph by starting from a pair of nodes and inserting any new node in-between a random existing pair of neighbors. However, as this algorithm requires passing IP addresses to other peers, it might be less resilient against misbehaving nodes, and the authors of Dandelion decided against it. Instead, the line graph is approximated in Dandelion [15]. Each node selects randomly k other nodes among its neighbors of the complete Bitcoin graph and asks them for their in-degree. An outgoing connection is built

up to the node with the lowest in-degree. The authors of Dandelion showed that using $k \geq 2$, the majority of nodes have a degree of 2, making the algorithm a valid approach for approximating a line graph in a fully-distributed and asynchronous way. In Dandelion++, the authors came to the conclusion that the interactivity required from asking about other nodes' in-degrees can be misused by malicious peers. An attacker could lie about its own in-degree and encourage other peers to connect to it. Such attacks can decrease anonymity since the adversary would likelier be the first hop for propagated transactions. As a result, the 4-regular graph is approximated non-interactively in Dandelion++. For the construction, every node picks two random neighbors of the complete graph and connects to them. The result is the approximated 4-regular graph, which optimizes the formal anonymity guarantees of Dandelion++ between the case of a known and the case of an unknown privacy-subgraph.

In the next section, the content discovery of Bitswap is adapted based on Dandelion's approach for transaction propagation. Content discovery requests are routed over a random walk to a proxy in order to obfuscate the source of the request. The random walk is performed on a privacy-subgraph that is constructed from the network graph which Bitswap operates on. The construction algorithm described in Dandelion++ is generalized for being able to approximate 2- and higher regular graphs.

4. Content Discovery through Random Walks

In this section, the privacy issues of Bitswap are addressed by proposing an adaptation of the Bitswap protocol that utilizes random walks to preserve privacy. The original Bitswap protocol is henceforth referred to as *Baseline Bitswap*, while the new protocol is denoted as *RaWa-Bitswap*. Exactly like Baseline Bitswap, RaWa-Bitswap aims for content discovery and data exchange in a P2P network. In contrast to Baseline Bitswap, RaWa-Bitswap employs a content discovery approach that preserves privacy of the nodes requesting content. In the following, an overview of RaWa-Bitswap is provided first. Next, the protocol's design is described in detail, and the taken decisions are discussed. RaWa-Bitswap is implemented for evaluation in Section 5. Finally, the characteristics of this implementation are presented.

4.1. Protocol Overview

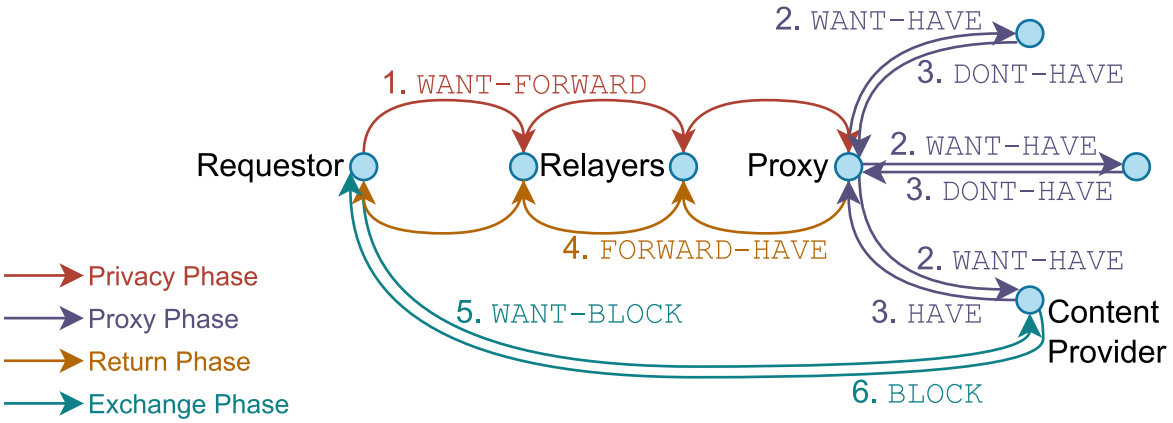

Figure 4: Message flow in RaWa-Bitswap.

For RaWa-Bitswap, the results of Dandelion [5] and Dandelion++ [18] are utilized by introducing a proxy into the Bitswap protocol. Instead of transactions, the interest in a CID is forwarded by the new message `WANT-FORWARD`. A `WANT-FORWARD` message is relayed through the network on a random walk to a proxy. Thus, when a node sends a `WANT-FORWARD`, the node can plausibly deny that itself is interested in the CID as it could be forwarding the interest of any other node. The proxy acquires one or more content providers. This provider information is routed back to the requestor in a `FORWARD-HAVE` message. After that, the content can be exchanged directly between the requestor and the provider. As in Dandelion, a privacy-subgraph is used for increased mixing of the forwarded messages and thereby achieving better privacy properties. The RaWa-Bitswap protocol is schematically visualized in Figure 4.

In the style of Dandelion, the protocol is divided into four phases: the privacy phase, the proxy phase, the return phase, and the exchange phase. The privacy phase starts

with a requestor constructing a WANT-FORWARD message that includes the root CID of the content the requestor wants to obtain. The requestor sends the WANT-FORWARD to a neighbor in the privacy-subgraph, which initiates a session for the requestor. Design and construction of the privacy-subgraph are discussed later in Section 4.2.7. The receiver of the WANT-FORWARD tosses a biased coin. With probability p, the receiver transitions into the proxy phase, and with probability $1 - p$, the receiver relays the WANT-FORWARD to another neighbor within the privacy-subgraph. The relaying is iteratively continued until one node eventually transitions into the proxy phase. Any relayer or proxy stores the combination of the peer from which they received the WANT-FORWARD and the corresponding CID. Relayers additionally store the peer to which they forwarded the message. This allows routing the response back to the requestor. When the requestor or a relayer forwards a message, they follow a strategy in choosing a neighbor for forwarding. The simplest strategy is to choose a neighbor at random. The forwarding strategies are explained in detail in Section 4.2.8.

When transitioning into the proxy phase, the receiver of the WANT-FORWARD message becomes a proxy. Initially, the proxy creates a special proxy session used to gather information about which peers may possess the block corresponding to the CID specified in the WANT-FORWARD message. To discover content providers for this CID, the proxy sends out a WANT-HAVE for this CID to all its connected neighbors in the same way as the initial query of Baseline Bitswap. For this, the complete P2P graph is used instead of the privacy-subgraph. Whenever a node responds with a HAVE message, the proxy session wraps the PeerID and the *multiaddresses* of the responding node in a FORWARD-HAVE message. A multiaddress is the data structure employed by Bitswap's networking interface for connecting to another peer and encapsulates a network address, for example an IP address and a TCP port, in a composable way [47]. In case no neighbor responds with a HAVE, the proxy queries the content routing subsystem for content providers of the CID. The provider information obtained from the content routing subsystem is then incorporated into a FORWARD-HAVE. In this case, the provider information only includes a PeerID (*i.e.*, no multiaddresses) for each provider, as this is the only information obtained by the content routing subsystem when queried for a CID. The proxy session sends any constructed FORWARD-HAVE message back immediately to the peer from which it received the initial WANT-FORWARD.

The return phase is the process of routing the FORWARD-HAVE messages back to the requestor. Using the stored information about the received WANT-FORWARD messages, the proxy and the relayers can send the message backward along the same path as the initial random walk.

The exchange phase starts when the requestor receives the first FORWARD-HAVE. The content provider information inside any received FORWARD-HAVE is added to the requestor session. Using this information, the session can directly send a WANT-BLOCK message for the root CID to a content provider. The content provider transmits the block in a BLOCK message, which contains the CIDs of the next level of the Merkle DAG. Any subsequent CIDs on the next levels of the Merkle DAG are directly requested from the same session peer that received the first WANT-BLOCK. The session tries to request all blocks from this single peer to limit exposure of the interest in the CID. Only if this

peer responds with a DONT-HAVE message, the requestor selects a new session peer for querying. In contrast to Baseline Bitswap, the requestor session of RaWa-Bitswap does not send WANT-HAVE messages to the session peers not selected for the WANT-BLOCK.

When sending the initial or any subsequent WANT-FORWARD message, the requestor session initiates an *unforwarded search timer*. If no FORWARD-HAVE message reaches the requestor in response to the WANT-FORWARD after a timeout, the requestor queries the content routing subsystem as a fallback. The resulting provider information is incorporated into the requestor session.

RaWa-Bitswap obfuscates the source of a WANT-FORWARD request, thereby preserving users' privacy during the content discovery. It narrows the data exchange down to the direct interaction between the requestor and ideally one content provider. Thus, the disclosure of interest in CIDs through WANT-BLOCK messages is limited. The reasoning for applying source obfuscation by the random walk only to the content discovery is further explained in the next section.

4.2. Protocol Design

The design of RaWa-Bitswap is primarily inherited from Dandelion and Dandelion++. However, further considerations are necessary in the design process, as Bitswap's message flow differs from Bitcoin's transactions in one crucial aspect. Whenever a client message is sent, it triggers a corresponding server message in response. Consequently, applying Dandelion's approach to Bitswap presents challenges. This section provides deeper insights into the design process of RaWa-Bitswap. To begin, the adversarial model that RaWa-Bitswap aims to address is described. Subsequently, the rationale for applying source obfuscation exclusively to content discovery is clarified. Finally, the individual components of RaWa-Bitswap are presented in detail: the messages, the requestor session, the relay management, the proxy session, the privacy-subgraph, and the forwarding strategies.

4.2.1. Adversarial Model

As most of RaWa-Bitswap's design is based on Dandelion++, its adversarial model is also adopted to ensure privacy preservation. The authors of Dandelion++ unified two adversarial models that had been studied in the context of privacy within the Bitcoin network. In [4] and [28], a single supernode adversary is proposed, capable of establishing outbound connections to as many nodes as it desires, up to connecting to all nodes in the network. The other model is that of Dandelion, which represents a botnet comprising numerous honest-but-curious peers. Each of these peers creates a limited number of outbound connections as specified by the protocol. The resultant adversarial model of Dandelion++ and the one adopted in this study is a botnet adversary capable of compromising a certain fraction of the network's nodes and establishing an unrestricted quantity of outbound connections.

This botnet adversary is modeled as a set of cooperating hosts distributed throughout RaWa-Bitswap's network. The adversary controls a fraction α of the network's nodes.

These nodes are not required to follow the protocol, and thus, they can establish an arbitrary number of connections to other nodes. The botnet eavesdrops on all messages that reach its nodes. Information about these messages and the benign nodes may be recorded, and insights into the network topology can be collected. By consolidating this knowledge, the adversary attempts to identify the content that honest users are interested in.

The overall goal of the adversary is mass detection of users' interests. All users' interests should be unveiled by linking the CIDs included in `WANT-HAVE`, `WANT-BLOCK`, and `WANT-FORWARD` messages to the node that actually seeks to download the corresponding block. The adversary attempts to link any observed message of these three types to the PeerID of the node that constructed the originating `WANT-FORWARD` for the CID. It is important to note that from the PeerID, the IP address (*i.e.*, multiaddresses) of a user can be accessed via the DHT in IPFS. Although an IP address is not directly associated with a human identity, it substantially limits the set of potential candidates [5]. In conclusion, due to this adversary's objective, RaWa-Bitswap aims to ensure network-wide privacy without necessitating an alteration of the users' behavior.

4.2.2. Content Discovery

RaWa-Bitswap addresses the objective of network-wide privacy by utilizing a random walk solely for content discovery. As seen in Section 4.1, content providers are routed back to the requestor in response to a `WANT-FORWARD` message. Instead of the `FORWARD-HAVE` response, `BLOCK` messages could be directly returned to the requestor. To achieve this, the proxy would need to acquire the requested block in place of the identity of the content providers. This approach would extend the mixing effect of the random walk to the data exchange process. The following discusses the issues associated with this strategy and highlights that it is not necessary to enhance privacy for Bitswap. Furthermore, the challenges faced by the content discovery approach are explored, along with the methods employed to address them. An overview of both approaches' challenges is provided in Table 1.

First and foremost, churn presents a potential challenge for both approaches. When relaying nodes depart from the network, it might become infeasible to return the outcome from the proxy back to the requestor. However, the impact of churn is more significant in the context of the data exchange approach. This is because files can be divided into multiple blocks, resulting in the need to route numerous messages back to the requestor, in contrast to the ideally single needed `FORWARD-HAVE` message in RaWa-Bitswap. Relaying multiple blocks consumes more time than relaying a single `FORWARD-HAVE` message, as the proxy not only has to discover providers but also acquire the blocks. Consequently, the likelihood of a relayer leaving the network during the proxy phase is higher when employing the data exchange strategy.

The impact of churn might be less under the content discovery approach, but it is still present. As a fallback, RaWa-Bitswap implements the unforwarded search timer that enables a requestor to query the content routing subsystem by itself, as soon as the timer triggers. Furthermore, as presented in Section 4.2.4, `WANT-FORWARD` messages

Approach	Challenge	Consequent design decision for RaWa-Bitswap
Data Exchange over random walk	Higher impact of churn	Use random walks solely for content discovery
	Increased load on the network	
Content discovery over random walk	Requestor likely unconnected to relayed providers	Return multiaddresses for each provider instead of solely a PeerID
	Privacy risk through requestor session requests	From the requestor session, send `WANT-BLOCK` messages only to a single peer and no `WANT-HAVE` messages

Table 1: Challenges of the two possible approaches for applying random walks to Bitswap in comparison.

can be retried along the same path as initially. In case any peer along the path is unavailable, the proxy phase is started at the first hop that encounters an unavailable successor.

When blocks are relayed, the network load significantly increases. A `WANT-FORWARD` message, which both approaches would incorporate, is relatively small as it only includes one CID. However, a `BLOCK` message can have a size of up to 2 MiB [42]. Furthermore, multiple instances of these messages need to be relayed, often traversing multiple edges of the graph in the data exchange strategy. This underscores the increased network load. In contrast, a `FORWARD-HAVE` contains, besides the CID, one or more content providers each represented as a PeerID. Therefore, a `FORWARD-HAVE` message is significantly smaller in size compared to most `BLOCK` messages. The drawbacks of the data exchange approach have led to the decision to apply source obfuscation through the random walk exclusively to content discovery.

Nevertheless, the content discovery approach also presents challenges. Section 2.2 declares that Bitswap's content discovery aims to find a PeerID of a content provider. However, if the provider information would solely be returned from the proxy to the requestor as PeerIDs, an issue would arise. In Baseline Bitswap, content discovery through `WANT-HAVE` messages identifies peers that are already connected to the requestor. In RaWa-Bitswap, the requestor is in most cases not yet connected to the providers received via `FORWARD-HAVE` messages. If only a PeerID is returned for each provider, the requestor would need to discover multiaddresses to connect to the content providers. In the case of IPFS, this would involve querying the DHT for the peer record to obtain one or more multiaddresses. This DHT query could become a performance bottleneck, contradicting Bitswap's goal of faster content resolution [63]. When the proxy acquires information about a provider through a `WANT-HAVE` message, *i.e.* not

through the content routing subsystem, the proxy is already connected to the provider and knows at least one multiaddress. As a result, the proxy includes the multiaddresses it knows in the FORWARD-HAVE messages, along with the PeerID for each provider. By using these multiaddresses, the requestor can directly connect to the providers without having to query the content routing subsystem.

In Baseline Bitswap, all peers in the requestor's session receive a WANT-HAVE message if they are not chosen for the optimistic WANT-BLOCK. This could potentially pose a privacy risk if carried over to RaWa-Bitswap. In most cases, the root CID is requested only from a proxy. Consequently, an adversary could reasonably assume that any WANT-HAVE containing a higher-level CID originates from the requestor itself. To mitigate this risk, RaWa-Bitswap refrains from sending WANT-HAVE messages from the requestor session. However, the interest in a CID is still revealed through WANT-BLOCK messages. In RaWa-Bitswap, a WANT-BLOCK message is exclusively sent by a requestor. Thus, any recipient of a WANT-BLOCK can be certain that the sender is the originator. This is deemed acceptable, as the protocol's objective is to ensure network-wide privacy, rather than focusing on the individual level. Nonetheless, the requestor session should restrict the number of WANT-BLOCK recipients and try to acquire all blocks from a single provider. Consequently, RaWa-Bitswap selects a new peer from the requestor session for the WANT-BLOCK requests only if the previous one has sent a DONT-HAVE message.

RaWa-Bitswap aims to download content from a single provider in the ideal scenario. It is worth mentioning that this design relies on the assumption that all blocks of a file or directory are typically stored on a single peer. This assumption is reasonable, because recent research pointed out that the majority of content in IPFS is provided by at most two nodes [7]. By narrowing down data exchange to interactions between ideally two nodes, RaWa-Bitswap reveals the interest in a CID through WANT-BLOCK messages to a minimum of nodes. Consequently, the protocol can enhance privacy by applying source obfuscation exclusively to the content discovery.

4.2.3. Additional Messages

While Baseline Bitswap's messages keep their semantics in RaWa-Bitswap, two additional messages are introduced. To facilitate the forwarding of content discovery requests, it is necessary to be able to distinguish between requests that should be forwarded and those that should be answered immediately. Although this could be accomplished by introducing a flag into the WANT-HAVE message, the protocol becomes more comprehensible with the introduction of the newly added client message WANT-FORWARD. To transmit the outcome of the proxy's content discovery through the network, there's a need to include provider information for a specified CID in a new server message. This is because a HAVE message can only announce that the sender of the message stores the content. Consequently, the new server message FORWARD-HAVE is employed for transferring this provider information. RaWa-Bitswap utilizes the same message envelope as Baseline Bitswap, containing one wantlist and multiple block presences. The only differences lie in the wantlist, which can contain multiple entries of the WANT-FORWARD message, and the block presences, which can include FORWARD-HAVE

messages. The definitions of the two new messages are provided in the following:

- `WANT-FORWARD`: Requests to acquire content providers for a specified CID. The receiver takes one of two actions: it either becomes a proxy with probability p or forwards the message with probability $1 - p$. Any discovered providers are sent back in `FORWARD-HAVE` messages.
- `FORWARD-HAVE`: Informs about one or more content providers for a specified CID. Each content provider is included by its PeerID and optionally one or more multiaddresses. It should be relayed to any neighbor that previously sent a `WANT-FORWARD` message for the same CID.

The messages of Baseline Bitswap are specified in the Protocol Buffers 3 format [21]. The new messages have been integrated into these Protocol Buffers. In Appendix A, the updated message specification can be found. For comparison, the Protocol Buffers of Baseline Bitswap can be located in its specification [42].

4.2.4. Requestor Session

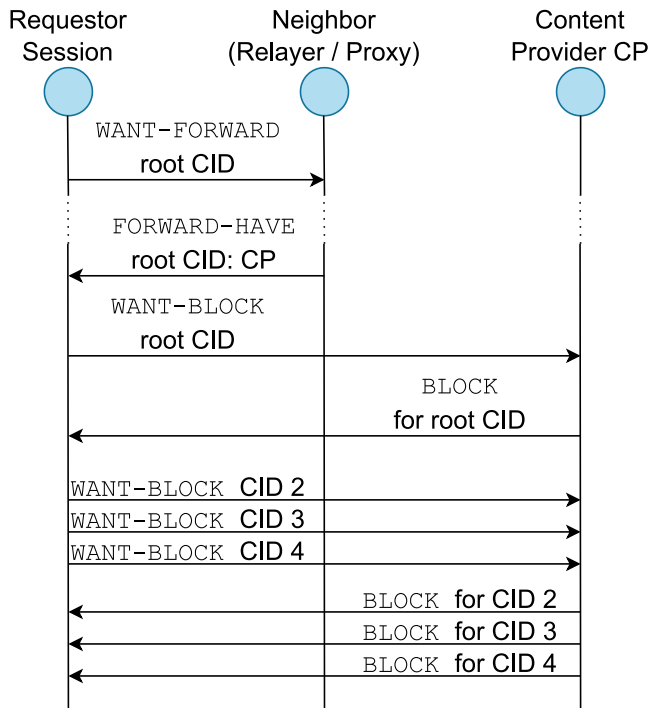

Figure 5: Message exchange from the viewpoint of the requestor session.

The requestor session in RaWa-Bitswap follows the design of Baseline Bitswap's session. The primary change is that whenever `WANT-HAVE` messages would be sent out to all neighbors, instead a `WANT-FORWARD` message for one CID is passed to one neighbor

within the privacy-subgraph. The requestor expects to receive content providers via a FORWARD-HAVE message in response, and these providers are added to the session. During the exchange phase, the requestor session attempts to acquire all blocks from the first content provider that was added to the session. It does this by sending WANT-BLOCK messages and traversing the Merkle DAG to request any subsequent CIDs. Figure 5 illustrates an example of the requestor session's message exchange in the ideal case.

When the requestor session receives a FORWARD-HAVE, it connects to the included providers. In case the multiaddresses of a provider are included, the requestor can directly establish a connection. If this is not the case, the requestor first queries the content routing subsystem for these addresses, delaying the download. The requestor builds up the connection to the provider once the content routing subsystem request has succeeded.

The requestor session incorporates various fallback strategies for fault tolerance. If the content provider currently queried by the WANT-BLOCK messages responds with a DONT-HAVE, a new provider is probabilistically selected from the session peers. This selected peer becomes the only target for subsequent WANT-BLOCK messages. Exactly like Baseline Bitswap, the probability of selection is proportional to the number of successfully received blocks from that peer. This approach also minimizes the number of different peers requested by WANT-BLOCK messages, as a previously successful provider is more likely to be chosen again later. The pruning of a session peer is also inherited from Baseline Bitswap and is performed after 16 DONT-HAVE messages from that peer. If all peers in the requestor session respond with a DONT-HAVE for a specific CID, the requestor passes a new WANT-FORWARD message for this CID to one neighbor. The requestor session also utilizes the idle tick and the periodic search timer, but in contrast to Baseline Bitswap, it retries a WANT-FORWARD message when these timers trigger. Due to churn in a P2P network, a relayer could leave the network before it passes a FORWARD-HAVE backward. Additionally, as described in Section 4.2.1, the adversarial model assumes that malicious nodes may not follow the protocol and could refuse to forward a request. To handle these cases, an *unforwarded search timer* is initiated whenever a WANT-FORWARD message is sent to a neighbor. After a timeout, the requestor session queries the content routing subsystem for the CID corresponding to the WANT-FORWARD that initiated the timer. Incorporating providers from the content routing subsystem enables the requestor session to continue normally in the aforementioned cases. Which duration u should be used for the unforwarded search timer is discussed in Section 4.3. An unforwarded search timer is stopped when a FORWARD-HAVE message is received for the respective CID before the timer triggers.

In conclusion, the requestor session performs content discovery by forwarding WANT-FORWARD messages and data exchange with ideally one peer through WANT-BLOCK messages. It resorts to querying the content routing subsystem only as a fallback when no response to a WANT-FORWARD message is received. Through its fallback mechanisms, the requestor session ensures fault-tolerant operation for retrieving the desired content.

4.2.5. Relay Management

Transmitting messages over a random walk requires a relaying functionality, which is not realized in Baseline Bitswap. De la Rocha *et al.* [14] explored message forwarding in Bitswap by introducing a *time to live* (TTL) parameter into `WANT-HAVE` and `WANT-BLOCK` messages to enhance performance. As long as a message has a TTL greater than 0, it should be forwarded to a configurable number of neighbors in addition to sending a response. Each forwarding node reduces the TTL by 1 and keeps track of the peer transmitting the message. Using this tracked information, nodes can relay `HAVE` and `BLOCK` messages back to the original requestor. This forwarding mechanism does not improve privacy, as the source of a request can still be inferred by analyzing the timing pattern of the requests [5].

The objective of RaWa-Bitswap differs from that of TTL-based forwarding. RaWa-Bitswap aims to transmit a message over a single path rather than flooding it through the network over a number of hops. Forwarding is adapted for this objective in RaWa-Bitswap. Only the newly introduced `WANT-FORWARD` and `FORWARD-HAVE` messages are being relayed for content discovery. The rationale behind the design choice against data exchange is discussed in Section 4.2.2, and the introduction of these new messages is explained in Section 4.2.3. Any recipient of a `WANT-FORWARD` message stores information about the message sender associated with the specified CID and forwards the message to another neighbor within the privacy-subgraph with a predefined probability. The neighbor to which the message is forwarded is also stored by a relayer. Two strategies for selecting the neighbor to forward to are presented in Section 4.2.8. The forwarding mechanism provides nodes with plausible deniability for the `WANT-FORWARD` requests, as any request could potentially be sent on behalf of another node. Leveraging the stored information about received `WANT-FORWARD` messages, nodes can send `FORWARD-HAVE` messages to the interested neighbors. Consequently, `FORWARD-HAVE` messages are routed backward along the same path as the initial request.

If a relaying node receives a `WANT-FORWARD` with the same CID from the same sender as earlier, the node forwards the message to the same neighbor as before using the stored relay information because this situation likely represents a retried message. In case the relaying node is the proxy for this sender-CID combination, it does not handle the forwarded message as the proxy session is still running. Only if the target peer for forwarding has disconnected, the relayer initiates the proxy phase for a retried message. This behavior ensures that `WANT-FORWARD` requests follow the same path as the initial random walk in case they are retried. This also limits the probability that the message is exposed to an adversary. For any new sender-CID combination that includes an already relayed CID, the relayer selects a different target that did not receive a forward request with this CID yet. In case no target is left, the relayer starts the proxy phase early. This method ensures that as soon as the forwarded message encounters a loop, it is not forwarded further to avoid routing messages in circles. In a simulation of an earlier version of the RaWa-Bitswap protocol, an unlimited number of `FORWARD-HAVE` messages could be triggered due to loops.

For any new `WANT-FORWARD` request, it is randomly decided at each hop whether to

transition into the proxy phase. Specifically, with a probability of p, a `WANT-FORWARD` receiver chooses not to forward the message and instead assumes the role of a proxy for the request. When a node becomes a proxy, it initiates a proxy session, which is discussed in the following section.

4.2.6. Proxy Session

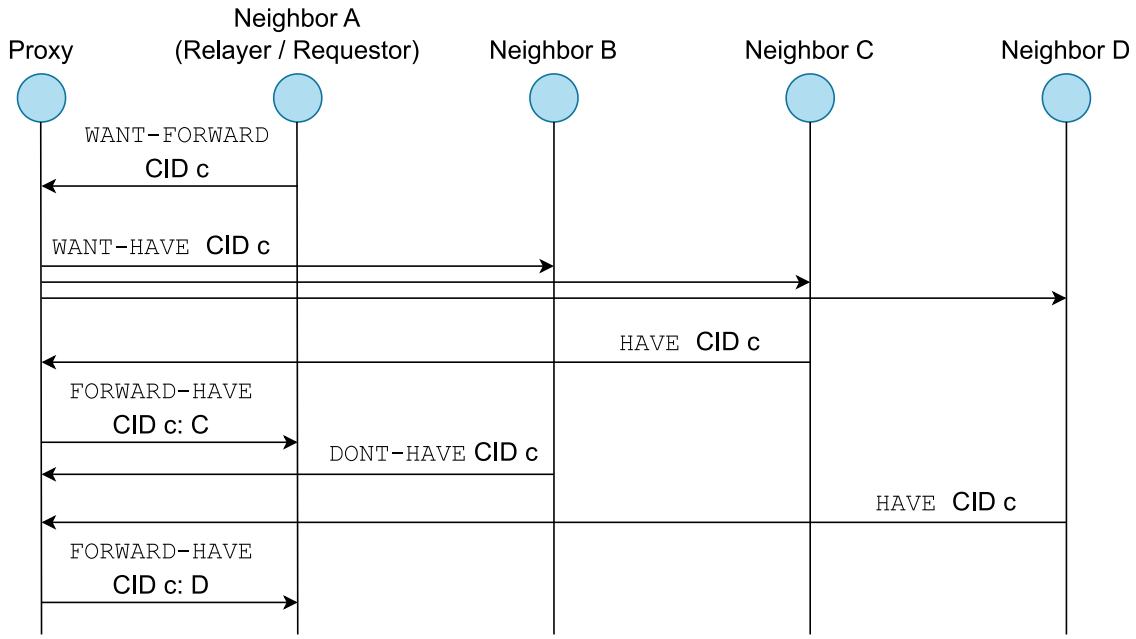

Figure 6: Message exchange from the viewpoint of the proxy session.

In RaWa-Bitswap, a proxy is responsible for identifying content providers for a CID by order of a `WANT-FORWARD` request. Any discovered provider is sent back to the sender of the request using the relay information. A proxy utilizes a proxy session to gather information about neighbors that may possess the requested content. The proxy session functions similarly to Baseline Bitswap's session but operates only on one CID and does not send `WANT-BLOCK` requests. Upon initialization, the proxy session passes a `WANT-HAVE` request to all connected neighbors to discover which ones store the block corresponding to the requested CID. Similar to Dandelion's spreading phase, this `WANT-HAVE` request is sent to all neighbors in the complete P2P graph (*i.e.*, not just the privacy-subgraph). If any peer responds with a `HAVE` message, the PeerID and multiaddresses of the possessing peer are included in a `FORWARD-HAVE` message, which is returned to the node that sent the `WANT-FORWARD` request. The proxy also initially checks if it stores the block itself locally. In such a case, it sends a `FORWARD-HAVE` message with its own PeerID and multiaddresses. An example of the message exchange from the proxy session's viewpoint is presented in Figure 6.

A proxy consults the content routing subsystem only if it was unsuccessful in the

initial discovery through WANT-HAVE messages. If the initial discovery was successful, the session shuts down before. Once all neighbors have responded with a DONT-HAVE, the content routing subsystem is queried for the requested CID. This design implies that every node can plausibly deny that it is interested in the CID of a request to the content routing subsystem because the node could also be a proxy. The PeerID of every discovered provider from the content routing subsystem is returned in a FORWARD-HAVE message. The proxy session also utilizes the 1-second idle tick from Baseline Bitswap. However, it can trigger the query to the content routing subsystem only once since the proxy session does not handle blocks. The proxy session is terminated after querying the content routing subsystem since it operates on a single CID. The periodic search timer is irrelevant for the proxy session, as the idle tick triggers earlier and eventually shuts down the session.

4.2.7. Privacy-Subgraph

By forwarding the WANT-FORWARD messages through the network, requests from different users move along the same links. Generally, mixing requests from different users should increase the difficulty for an adversary to identify the originator. With a smaller node degree, more messages flow over the same link, increasing the mixing effect. Therefore, the authors of Dandelion introduce the directed privacy-subgraph into their protocol [5]. As the privacy-subgraph, Dandelion assumes a 2-regular graph, and Dandelion++ assumes a 4-regular graph. The usage of a privacy-subgraph is adopted in RaWa-Bitswap. The privacy-subgraph comes into play during the privacy phase when WANT-FORWARD messages are forwarded, and during the return phase when FORWARD-HAVE messages are relayed along the same path in the reverse direction. To be precise, the privacy phase uses a directed subgraph derived from the complete P2P graph, and the graph used during the return phase is the transpose of this directed subgraph. In the following, it is discussed which structure of the privacy-subgraph would provide the best privacy enhancement.

Sharma *et al.* [57] proposed a Bayesian framework for evaluating the anonymity of P2P network schemes and applied it to Dandelion and Dandelion++. They used the entropy of potential transaction senders as an anonymity metric and found that the anonymity of both protocols is limited. By running simulations using the usual Bitcoin graph (*i.e.*, a 16-regular graph) in place of the privacy-subgraph, Sharma *et al.* demonstrated that increasing the node degree in the privacy-subgraph leads to better anonymity for Dandelion and Dandelion++. In contrast, Dandelion and Dandelion++ used precision and recall as the anonymity metrics and provided theoretical bounds for these metrics. Their formal anonymity guarantees are based on these theoretical bounds. The reason for these different findings is that Sharma *et al.* assumed that the privacy-subgraph is known in their model. They claimed that an adversary could easily learn the subgraph and provided an algorithm for this purpose. The idea of the algorithm is as follows: When an adversary is connected to all benign nodes, it can forward multiple transactions to one honest node and measure the frequency with which each node diffuses these transactions. With this information, the adversary can

infer the honest node's successors because the distribution of the diffusion frequency per node depends on the probability of transitioning into the spreading phase, and thus, closer nodes are likely to diffuse more transactions. By repeating this process for all honest nodes, the adversary can derive the complete privacy-subgraph. To prevent graph learning attacks, Dandelion and Dandelion++ proposed to reconstruct the privacy-subgraph every 9 minutes, making it dynamic compared to Bitcoin's relatively static graph. However, Sharma *et al.* did not consider the frequent reconstruction of the privacy-subgraph in their evaluation. It is unclear if an adversary is able to employ the learning strategy in a large P2P network like the Bitcoin network, with around 10,000 different nodes being active per day [17]. It takes a noticeable amount of resources and time to connect to all network nodes and send multiple (they proposed 50) transactions per node. Sharma *et al.* also did not take the strategy's effort into consideration for their results. They simply assumed the privacy-subgraph as publicly known. In Dandelion++, the authors no longer assumed an unknown privacy-subgraph and pointed out that it is hard to evaluate how likely it is for an adversary to infer the complete graph. Therefore, they optimized the protocol between the cases of the subgraph being known and unknown by utilizing a 4-regular graph. This approach is derived from their analysis of the known subgraph case, which showed that an increased node degree results in better anonymity properties, a finding later confirmed by Sharma *et al.*. The assumptions and privacy-subgraph structures proposed by the three studies are summarized in Table 2. In conclusion, the previous arguments imply uncertainty about whether an adversary would be capable of learning the complete privacy-subgraph and if assuming a known graph is reasonable.

Study	Assumption about privacy-subgraph	Structure of privacy-subgraph which gives the best anonymity properties
Dandelion [5]	Unknown	2-regular graph
Dandelion++ [18]	Unclear if adversary can learn the graph quickly enough	4-regular graph, which optimizes between the two extreme assumptions
Sharma *et al.* [57]	Known	Complete Bitcoin graph (*i.e.*, a 16-regular graph)

Table 2: Comparison of the investigations of Dandelion's privacy-subgraph.

Due to the uncertainty regarding the practicality of learning a dynamic privacy-subgraph, determining which structure would offer the highest privacy enhancement for RaWa-Bitswap is not straightforward. Consequently, all three aforementioned options are further investigated in this work. The target out-degree per node η is introduced into RaWa-Bitswap as a parameter, enabling the configuration of the privacy-subgraph's structure. Let $\Delta(G)$ denote the highest node degree within graph G. With G being the complete network graph, values of $\eta \in \{1, 2, \Delta(G)\}$ are tested in Section 5, evaluating

Algorithm 1 Approximate 2η-Regular subgraph from network graph

Input: Complete network graph $G(N, E)$, target out-degree η per node
Output: A directed graph $S(N, F)$ with average degree 2η

```
F ← ∅
for all n ∈ N do
    M ← {m | {n, m} ∈ E}
    O ← ∅
    for all i ∈ {1, ..., η} do
        o ~ Unif(M \ O)
        O ← O ∪ {o}
    end for
    F ← F ∪ {n → o | o ∈ O}
end for
return S(N, F)
```

the use of a 2-regular and a 4-regular subgraph, as well as the complete network graph. Similar to Dandelion, constructing a regular graph in RaWa-Bitswap poses challenges due to the distributed nature of a P2P system. In the following sections, an algorithm capable of approximating 2- and 4-regular (and higher degree) privacy-subgraphs in RaWa-Bitswap is proposed.

As the privacy-subgraph is created in a P2P system, the construction algorithm should be asynchronous and distributed. The construction should not reveal the topology to prevent adversaries from gaining an advantage in identifying the originator of a request. Dandelion and Dandelion++ each provide a relatively simple algorithm for this purpose. While Dandelion's line graph approximation requires interactivity of nodes, in Dandelion++, the authors notice that this interactivity can be exploited by an adversary. Therefore, Dandelion++'s subgraph construction algorithm is adopted for RaWa-Bitswap and generalized to approximate any directed regular graph, as shown in Algorithm 1.

Given the complete network graph G with the set N of nodes and the set E of edges, Algorithm 1 outputs, for every node, a number η of random neighbors as its successors in the subgraph. For every node n, the set of successors O is determined. The set M denotes all neighbors of the current node n in graph $G(N, E)$. In every round $i \in \{1, \ldots, \eta\}$ for the current node n, the algorithm selects another successor o randomly under uniform distribution from the not yet chosen neighbors $M \setminus O$. For every successor, a directed edge $n \to m$ is added to the set F of subgraph edges. The returned graph S has the original set N of nodes with a set of directed edges F.

The constructed graph is not an exact regular graph, but its expected node degree is $d = 2\eta$, assuming $2\eta \leq \delta(G)$. If $2\eta \geq \Delta(G)$, the algorithm selects all neighbors of the complete graph, resulting in the same graph as the input. It should be noted that Dandelion's construction algorithm asks for the in-degree of nodes to make leaf nodes in the subgraph significantly less likely. Leaves are avoided because they can reduce the privacy properties of the resulting graph [5]. With $\eta \geq 2$, the approximated graph

of Algorithm 1 has no leaves, as it creates at least two outbound connections per node. With $\eta = 1$, leaves are likely to be present, potentially degrading the privacy-enhancing effects.

Following the proposal of Dandelion, a subgraph reconstruction timer is introduced in RaWa-Bitswap to add dynamicity and increase the difficulty of learning the subgraph. An adversary should not be able to learn a significant proportion of the subgraph within a certain timeframe. The authors of Dandelion assume a rate of 3 transactions per second and a Bitcoin network size of about 5,500 servers. They calculate that using a timer of 9 minutes, a reasonable botnet adversary can never learn more than 40 % of the nodes' positions. Karapapas *et al.* [26] observed approximately 20 % of Bitswap requests in the IPFS network over 24 hours. They monitored 49,155 files being requested, which provides an estimate of 2.84 Bitswap requests per second for the entire network. In a study by Balduf *et al.* [2], a conservative assumption of an average of 14,411.42 peers for the IPFS network size was found. Given these request and network size numbers, the 9-minute reconstruction timer appears more than suitable for RaWa-Bitswap. Each RaWa-Bitswap node should execute Algorithm 1 every 9 minutes. Since the algorithm operates asynchronously, the reconstruction does not need to be synchronized across nodes.

Losing a connection might pose an issue for returning `FORWARD-HAVE` messages to a requestor. However, reconstruction does not entail the loss of the connection to the predecessor. Disconnection only occurs when the connection is also lost in the complete network graph. In IPFS, connection management is a relatively complex process and does not simply mirror the buckets of the Kademlia DHT. For instance, a node might randomly terminate connections older than 30 seconds when the upper connection limit is reached [23, 41]. Additionally, as mentioned earlier, churn can be problematic for returning `FORWARD-HAVE` messages. IPFS's graph is highly dynamic, as observed in the study by Henningsen et al. [23]. They found that in 44 % of cases, a node leaves the network in less than 5 minutes after joining. In the event of a lost `FORWARD-HAVE` message, the requestor session's idle tick comes into play. After 1 second without a response, the idle tick triggers a second `WANT-FORWARD` message, forwarded over the same path as the initial random walk. If any node along the path went offline, the predecessor of the offline peer takes the role of the proxy when it receives the retried `WANT-FORWARD` message. This ensures the privacy-subgraph can be reliably employed for increased message mixing while making it challenging to learn.

4.2.8. Forwarding Strategies

When forwarding a `WANT-FORWARD` message, a node selects one of its successors in the privacy-subgraph for sending the message. While Dandelion++ does not propose it, it is possible to implement a strategy for selecting a successor for forwarding. It should be noted that when using a 2-regular subgraph, the choice of a forwarding strategy becomes irrelevant because a node only has one successor to select, and the messages follow a linear path when forwarded. In a 4-regular graph, one of the two successors can be chosen, and in a complete network graph, any neighbor may be selected. In the

following, two simple forwarding strategies are proposed, but more complex strategies could also be considered in future studies.

Random Forward: The node chooses one of its successors at random using a uniform distribution. In Dandelion++, transactions are forwarded to a random neighbor because it promises the highest privacy enhancement. Therefore, only this strategy is used in the evaluation in Section 5.

Closest PeerID Forward: IPFS's Kademlia DHT uses the XOR metric to order CIDs and PeerIDs by distance. When requesting the DHT, the number of queries to other peers depends on the proximity of the requestor to the queried key. With this strategy, a peer selects the neighbor which is closest to the CID of the `WANT-FORWARD` message according to the XOR metric. This ensures that a proxy is selected that is on average closer to the CID than using the Random Forward. Thereby, the number of DHT queries is reduced, and the proxy can more quickly respond with a `FORWARD-HAVE` in case the proxy has to query the DHT for discovering peers. This strategy is designed for IPFS, as other protocols integrating Bitswap may use another content routing subsystem. While the neighbor is not chosen at random with this strategy, the result is still a random walk since the length of a message's path is given at random by the proxy transition probability p. This strategy has the potential to reduce privacy drastically as the neighbor is chosen deterministically.

The forwarding strategy is one of the parameters that are left open for configuration. Next, all of these configurable parameters are summarized.

4.3. Protocol Parameters

RaWa-Bitswap employs multiple configurable parameters. This section presents all of them together and proposes sensible values for them. A summary can be found in Table 3.

The parameter p defines the proxy transition probability during the privacy phase. Any node receiving a `WANT-FORWARD` message tosses a biased coin. With probability p, the node transitions into the proxy phase. If a node does not start the proxy phase, it forwards the `WANT-FORWARD` to another neighbor within the privacy-subgraph. Generally, a lower value of p should increase privacy and decrease performance because a message is forwarded further through the network, making it more difficult for an adversary to track the message's path. With p, the number of hops a message is forwarded can be estimated. The probability X_p that a message is proxied within e hops under proxy transition probability p can be calculated from the complementary event:

$$X_p(e) = 1 - (1 - p)^e \tag{1}$$

Considering the bounds, with $p = 0$, the proxy phase would never start, and with $p = 1$, an adversary gains a significant advantage in detecting the message originator, as the first hop would always be the proxy for a `WANT-FORWARD`. Dandelion++ [18]

Parameter	Description	Proposed values
Proxy transition probability p	Defines how likely a node receiving a `WANT-FORWARD` message becomes a proxy instead of forwarding the message further.	$p \in [0.05, 0.3]$
Unforwarded search timer duration u	Determines the number of seconds after which a requestor queries the content routing subsystem as a fallback.	$u \in [2, 10]$
Privacy-subgraph's target out-degree η	Defines how many connections of the complete graph G each node should take for creating an edge in the privacy-subgraph.	$\eta \in \{1, 2, \Delta(G)\}$
Forwarding strategy	Determines which successor of the privacy-subgraph is chosen for forwarding a `WANT-FORWARD` message.	Random Forward, Closest PeerID Forward

Table 3: Configurable parameters of RaWa-Bitswap.

also utilizes a probability for deciding if a transaction should be diffused instead of forwarding it further. In the following, this probability is denoted as p_D. The authors of Dandelion++ show that the smaller p_D is, the lower the precision of an attacker for detecting the transaction originator. Therefore, they propose using low values for the probability. Any fraction of adversarial nodes poses the lowest detection precision when $p_D = 0$. They show that with $p_D \leq 0.2$, the precision increase compared to the case of $p_D = 0$ is limited to 0.1. It should be noted that it can make sense to deviate from this result to improve the performance of the protocol. Under Equation 1, with $p = 0.2$, a forwarded message is proxied in at least 90 % of the cases within 11 hops, in at least 95 % of the cases within 14 hops, and in at least 99 % of the cases within 21 hops. Analogously, with $p = 0.3$, a message is proxied in at least 90 % of the cases within 7 hops, in at least 95 % of the cases within 9 hops, and in at least 99 % of the cases within 13 hops. $p = 0.3$ can significantly reduce the expected number of hops, and thus enhance performance. Conclusively, a value of $p \in [0.05, 0.3]$ is proposed in this work.

The unforwarded search timer is introduced as a fallback in case a requestor does not receive a response to its sent `WANT-FORWARD` messages. When it times out, the

requestor session queries the content routing subsystem for the CID corresponding to the `WANT-FORWARD` that initiated the timer. When the requestor itself queries the content routing subsystem, it might reveal its interest in a CID to many peers, as discussed in Section 2.2. Therefore, queries to the content routing subsystem by the requestor itself should be avoided. As the idle tick triggers after 1 second without a block and initiates a second `WANT-FORWARD` for mitigating churn, the duration u of the unforwarded search timer should be considerably higher. To ensure waiting for the response to at least two `WANT-FORWARD` messages, the duration of the unforwarded search timer should be set to at least double the 1-second duration of the idle tick. Setting a high value for u can drastically reduce performance when the network undergoes high churn. For this reason, 10 seconds is proposed as an upper limit for u.

The target out-degree per node η defines the number of edges each node establishes in the privacy-subgraph. η serves as an input for Algorithm 1. Based on the discussion in Section 4.2.7, a 2-regular and a 4-regular subgraph, as well as the complete network graph G should be tested for the privacy-subgraph. Therefore, the target out-degree per node is $\eta \in \{1, 2, \Delta(G)\}$.

A node chooses a successor of the privacy-subgraph according to a strategy when forwarding a `WANT-FORWARD` message. This forwarding strategy might be the Random Forward, selecting a random successor, or the Closest PeerID Forward, selecting the neighbor with the PeerID closest to the CID of the message. As discussed in Section 4.2.8, the Random Forward offers better privacy and should be preferred over the Closest PeerID Forward when aiming for high privacy of the protocol.

It is worth noting that three timers each have a fixed duration for the evaluation, and these can also be considered as protocol parameters. To increase the difficulty of learning the privacy-subgraph, the subgraph is reconstructed frequently. Therefore, each node in RaWa-Bitswap runs Algorithm 1 repeatedly when the reconstruction timer triggers. The timer can be set on each node without synchronization, as the algorithm works asynchronously. As discussed in Section 4.2.7, inheriting the 9-minute duration from Dandelion for the reconstruction timer is sensible in the context of Bitswap. Furthermore, the timers from Baseline Bitswap are adopted and keep their default duration. The idle tick triggers when no block has been received within 1 second after session initialization or after an unusually long time since the last received block. In the requestor session, the idle tick forwards a `WANT-FORWARD` message again, and in the proxy session, it can run a content routing subsystem query once. The periodic search timer triggers every 60 seconds. It may only trigger in the requestor session, and there it also retries a `WANT-FORWARD` message. The fixed parameters are summarized in Table 4.

Different values of the configurable parameters and their impact on privacy and performance are evaluated using a simulated environment in Section 5. For this simulation, RaWa-Bitswap is implemented, which is shown next.

Parameter	Description	Value
Privacy-subgraph reconstruction timer duration r	Defines the frequency in seconds of each node running Algorithm 1 for constructing the privacy-subgraph.	$r = 540$
Initial idle tick duration t	Determines the initial number of seconds after which a `WANT-FORWARD` is retried in the requestor session, or the content routing subsystem is queried in the proxy session.	$t = 1$
Periodic search timer duration s	Defines the frequency in seconds at which a random `WANT-FORWARD` is retried in the requestor session.	$s = 60$

Table 4: Fixed parameters of RaWa-Bitswap.

4.4. Implementation

RaWa-Bitswap is implemented by modifying the reference implementation of Bitswap in Go, which is part of the Boxo [38] collection of various libraries for IPFS. For the modification, the Bitswap implementation of Boxo v0.8.0 is used as a base. The source code of RaWa-Bitswap is publicly available on GitHub[1].

To highlight the most significant change in the existing packages of the Bitswap reference implementation, the `Session` structure is adapted to make the functionality of both a requestor and a proxy session available. When blocks are requested, a `Session` is initiated with a flag declaring it as a requestor session. Furthermore, the message structure is modified according to RaWa-Bitswap's message specification, as shown in Appendix A. De la Rocha *et al.* [13, 14] implemented a `RelaySession` for their TTL-based forwarding approach. The `RelaySession` keeps track of the peers that sent `WANT-HAVE` messages and queries all neighbors for the corresponding CIDs. This `RelaySession` serves as a template for the implementation of a `RelayManager`, which only tracks the peers that sent `WANT-FORWARD` messages and does not query neighbors. Instead, it starts a proxy session with proxy transition probability p for each new `WANT-FORWARD` message, or with probability $1 - p$ forwards the message to another neighbor. Again, a flag declares a `Session` as a proxy session. A proxy session acquires

[1] `https://github.com/manuelwedler/boxo`

content providers and calls back the `RelayManager` when it finds some in order to send `FORWARD-HAVE` according to the tracked relay information. A `forwardGraphManager` structure is introduced into the `peermanager` package. The `forwardGraphManager` chooses a node's successors for the privacy-subgraph depending on a specified target out-degree. When forwarding a `WANT-FORWARD` message, the `forwardGraphManager` selects a node for forwarding according to a pre-configured strategy. The package `forwardstrategy` is implemented, which contains the forwarding strategies described in Section 4.2.8 and can be used with the `forwardGraphManager`. All newly introduced parameters, including the proxy transition probability, the duration of the unforwarded search timer, the privacy-subgraph's target out-degree per node, and the forwarding strategy, are made configurable when instantiating RaWa-Bitswap. In the implementation of Baseline Bitswap, blocks are immediately sent in response to `WANT-HAVE` messages in case they are below 1024 Bytes by default. Sending blocks in this edge case is removed from the RaWa-Bitswap implementation, because only a proxy sends `WANT-HAVE` messages, and the proxy does not care about the block.

Through obfuscating the source of content discovery requests, RaWa-Bitswap poses better privacy than Baseline Bitswap. However, forwarding can have an impact on performance [14]. To quantify the privacy and performance differences of both protocols, they are evaluated in a simulated environment. The implementation of this section is used for the evaluation, which is presented next.

5. Evaluation

To explore the privacy and performance properties of RaWa-Bitswap, this section proposes a testing setup. With this setup, simulations of both RaWa-Bitswap and Baseline Bitswap are conducted, and their privacy and performance properties are measured. The results are discussed at the end of this section, and directions are shown for future work.

5.1. Testing Setup

RaWa-Bitswap's goal is to enhance privacy compared to Baseline Bitswap. To quantify if and to what extent this is achieved, a measurement of privacy is conducted in a simulated environment. As the introduced forwarding may influence the performance of the protocol [14], RaWa-Bitswap's performance should also be measured. RaWa-Bitswap is evaluated using a simulated environment rather than proving theoretical bounds as in Dandelion and Dandelion++, because sensible theoretical bounds are hard to define for a dynamic network graph [18]. The authors of Dandelion and Dandelion++ assume a static graph for Bitcoin's complete graph. As discussed in Section 4.2.7, IPFS's graph is highly dynamic [23].

Simulations are conducted using the RaWa-Bitswap implementation of Section 4.4 in a Testground [61] setup. The Testground platform enables testing, benchmarking, and simulating P2P systems. The used version of Testground is v0.6.0. The simulations are run on a 64-bit Ubuntu 22.04.3 LTS system with Linux kernel 6.2.0-33-generic and Intel Core i7-6700K CPU with 16 GiB RAM. In all tests, each node is executed inside its own Docker container.

The creators of IPFS provide a Testground test plan for measuring the performance of Bitswap [43]. This test plan serves as the foundation and is further adapted for the simulations in this work. The Testground test plans can be found in the GitHub repository of RaWa-Bitswap[2]. Varying values of RaWa-Bitswap's configurable parameters are tested in the simulations. Specifically, these include the proxy transition probability p, the unforwarded search timer duration u, and the target out-degree per node η for the privacy subgraph. Section 4.3 proposes reasonable values which are utilized for these parameters in the simulations. The forwarding strategy remains consistent across all simulation runs. All nodes adhere to the Random Forward strategy because it avoids introducing determinism to the protocol, unlike the Closest PeerID Forward strategy, and therefore offers enhanced privacy properties.

An adversary is introduced into the test network, which attempts to identify for each honest peer which CID it is interested in. The privacy is measured by calculating *precision* and *recall* of the adversary's classification of the peers' interests. Three types of adversaries, each following distinct classification strategies, are implemented and tested. One of these adversaries has knowledge of the topology of the privacy-subgraph, while the others do not. Performance is quantified by measuring the time it takes for

[2] `https://github.com/manuelwedler/boxo/tree/main/testplans/rawa-bitswap`

nodes to fetch a block, denoted as the *time-to-first-block* (TTFB). For comparison, the tests for both metrics are also conducted against Baseline Bitswap in the version implemented in Boxo v0.8.0. The test network, the simulated adversaries, and the metrics for measuring privacy and performance are discussed in detail in the following sections.

5.1.1. Network

The same network setup is used for all simulations. It consists of 50 peers, each of which connects to 4 other nodes randomly selected under the condition that two peers never select each other. Therefore, the expected node degree is 8, and an 8-regular graph is approximated. A fraction α of the 50 peers is controlled by an adversary. Only the malicious nodes follow a different strategy for connecting to other peers, as explained in Section 5.1.3. At the beginning of each simulation run, each peer chooses 4 new non-adversarial nodes to connect to at random. The network comprises 50 peers, as it is the largest network that can be simulated with the aforementioned hardware. This network is sufficient to demonstrate differences in privacy and performance between RaWa-Bitswap and Baseline Bitswap, and between different parameter configurations within RaWa-Bitswap. 4 outbound connections per node are chosen because the complete network should have a significantly higher node degree than the smallest regular graph approximated for the privacy-subgraph, which is a 4-regular graph. Each network link has a latency of 100 milliseconds with a jitter of 10 %. The bandwidth of each link is 1 MiB per second.

The Bitswap implementation (*i.e.*, either RaWa-Bitswap or Baseline Bitswap) runs on top of a libp2p [44] host in the simulations. Libp2p is the P2P networking library on which Bitswap and IPFS are built. Each node runs one of these Bitswap instances with a TCP and a QUIC multiaddress. When instantiating Bitswap, a content routing subsystem is required. Using a real Kademlia DHT for the content routing subsystem would bloat the simulation and introduce significant complexity. Therefore, a dummy DHT is designed and utilized as the content routing subsystem. The dummy DHT resolves each query with a 100 % probability of success after a delay. A delay of 622 milliseconds, with a random variation of up to 10 %, is used. This delay is chosen because 622 milliseconds represents the median duration of a DHT query in IPFS [63].

Each simulation executes the following scenario: Every honest node generates a different random block of 150 kiB and stores the block locally. Afterwards, every honest node queries its Bitswap instance for the CID of one random block generated by another node. The nodes select the CID independently from each other; thus, a single CID might be queried by more than one node. All nodes run the query concurrently. The privacy and performance metrics are measured during this scenario. Notably, in this network configuration, nodes are able to download the block within a fraction of a minute. Consequently, the reconstruction timer never triggers. This scenario better resembles a realistic situation, which is expected to be busier than just one node downloading a file. Since all honest nodes request content, multiple messages for different CIDs, with varying nodes as the originators, are contributed to the network.

This diversity is essential for the privacy metric, which is explained in the next section.

5.1.2. Privacy Metric

Derived from the adversarial model described in Section 4.2.1, RaWa-Bitswap's goal is to provide network-wide privacy, rather than privacy on the individual level. Similar to Dandelion, the degree of network-wide privacy that RaWa-Bitswap achieves can be quantified by introducing a classification problem. When the requestor session tries to acquire content, it initially sends a `WANT-FORWARD` message for a CID. Subsequent `WANT-HAVE`, `WANT-BLOCK`, and `WANT-FORWARD` messages for the same CID may be observed in the network following this originating `WANT-FORWARD`. An adversary seeks to link any CID observed from `WANT-HAVE`, `WANT-BLOCK`, and `WANT-FORWARD` messages to the PeerID of the node that created the originating `WANT-FORWARD` for this CID. This linking can be interpreted as the classification of PeerIDs as interested in specific CIDs. Hence, the success of an adversary can be evaluated by calculating *precision* and *recall* as metrics for this classification problem.

Let H denote the set of all non-adversarial PeerIDs in the network, the set C consist of all queried CIDs in one simulation run, and $I = \{(h, c) \in H \times C\}$ be the set of all theoretically possible interests. As every honest node requests exactly one CID in the simulations, an adversary maps every honest PeerID in the network to an observed CID. Thus, there is the adversary's map function $M\colon H \to I$ that associates every honest node with an interest. The indicator function $\mathbb{1}\colon I \to \{0, 1\}$ checks if an adversary maps a node to the correct interest. The indicator function $\mathbb{1}$ outputs 1 if a given interest $i = (h, c)$ is a correct interest, meaning that node h is truly interested in the CID c, and 0 otherwise. Precision and recall are defined per honest node $h \in H$. Let $i_h = (h, c_h) \in I$ be the interest of h. An h mapped to the correct i_h is called *true positive*, and one mapped to a wrong $i \in I \setminus \{i_h\}$ is called *false negative*. A *false positive* represents any $g \in H \setminus \{h\}$ that is mapped to the interest (g, c_h) with the same CID as the node's interest. $K_h = \{(g, c) \in I \mid M(g) = (g, c) \wedge c = c_h\}$ denotes the set of all mappings to the CID of the node's interest. For an honest node h, precision $D(h)$ and recall $R(h)$ are defined as the following formulas:

$$D(h) = \frac{|\text{true positives}|}{|\text{true positives }| + |\text{false positives}|} = \frac{\mathbb{1}(M(h))}{|K_h|} \tag{2}$$

$$R(h) = \frac{|\text{true positives}|}{|\text{true positives }| + |\text{false negatives}|} = \mathbb{1}(M(h)) \tag{3}$$

If the denominator of Equation 2 is 0, $D(h) = 0$ is in effect. Since network-wide privacy should be measured, and not privacy of individual nodes, Equation 2 and Equation 3 are averaged for the entire network. The network-wide precision D and recall R can be calculated as follows:

$$D = \frac{\sum_{h \in H} D(h)}{|H|} \tag{4}$$

$$R = \frac{\sum_{h \in H} R(h)}{|H|} \tag{5}$$

Recall is the probability of an adversary detecting the interest of a node. It indicates how accurately an adversary can identify an individual interest. Precision decreases the more frequently a single CID is assigned. This means for a node that, even though it is correctly classified, the adversary's precision against it is less than 1. Therefore, it can be interpreted as the degree of a node's plausible deniability, with low precision indicating high plausible deniability. In the scenario studied here, a node whose requested CID is assigned more than once can deny being the source of the request, as it could have forwarded the request. Precision captures the adversary's ability to observe the interests of all network nodes and thereby identify one node's interest, providing a network-wide perspective. In conclusion, precision and recall collectively reflect the goal of network-wide privacy.

The classification problem described in this section implies an assumption: The adversary knows the PeerID of all participating nodes in the network since it classifies all these PeerIDs. This assumption is reasonable as there exists an IPFS crawler capable of collecting 50,000 distinct PeerIDs in about 4 minutes, on average [24]. When measuring privacy in the conducted simulations, a fraction α of the 50 network nodes act as an adversary. The various simulated adversaries and their strategy for tracking peer interests are defined next.

5.1.3. Simulated Adversaries

Three types of adversaries are introduced into the simulation scenario to quantify precision and recall. Each adversary associates observed CIDs with peers suspected of being interested in those CIDs. In this section, a basic *First-Spy Estimator* is introduced to illustrate the differences in precision and recall between RaWa-Bitswap and Baseline Bitswap. A vulnerability of RaWa-Bitswap is revealed, and a simulated adversary exploiting this vulnerability is described. In the classification process, an adversary with knowledge of the privacy-subgraph may gain an advantage. Therefore, the latter adversary is further enhanced with a simple heuristic that leverages the knowledge of the subgraph.

The simplest adversary tested is the *First-Spy Estimator* (FSE). The FSE runs one Bitswap instance that connects to every other peer in the network, in contrast to honest nodes which only connect to 4 peers. With one malicious node, α is 0.02 for the simulated network. In other studies, an FSE typically maps a request to the node from which it first observed the request [5, 12, 19]. In the classification problem studied here, PeerIDs are mapped to a CID because every peer is interested in exactly one CID,

and multiple peers might be interested in the same CID. Thus, the FSE used in the simulations associates every PeerID with the earliest observed CID from this peer that was not observed from another peer before. Any PeerID that is not assigned a CID after applying this strategy is mapped to a randomly selected CID observed during the simulation run. The goal of introducing this adversary is to demonstrate that a simple adversary can easily achieve high precision and recall on Baseline Bitswap, while on RaWa-Bitswap, more sophisticated approaches are required.

The forwarding approach of RaWa-Bitswap protocol introduces a vulnerability, that can reveal the originator of an observed `WANT-FORWARD` message. Assuming that an adversary can craft arbitrary messages, it can trigger a requestor to send `WANT-BLOCK` messages to the adversary. In RaWa-Bitswap, `WANT-BLOCK` messages are only sent from the originator of a request, allowing the adversary to directly identify the peer interested in the CID of a `WANT-BLOCK`. Whenever an adversary observes a `WANT-FORWARD` message, it immediately responds with a `FORWARD-HAVE` that includes the PeerID and multiaddresses of a malicious node itself. As a result, the requestor assumes that the malicious node stores the desired block and sends a `WANT-BLOCK` message to the adversary. This type of adversary is introduced into the simulations and is denoted as the *Want-Forward Exploiter* (WFE). It controls a fraction α of the network's RaWa-Bitswap nodes. Each malicious instance connects to a distinct set of 4 benign nodes. To connect to all honest peers, α is set to 0.2 throughout the simulation runs, resulting in 10 out of 50 nodes being malicious. The WFE collects data from all its nodes and associates every peer with the CID of the first `WANT-BLOCK` received from that peer. If no `WANT-BLOCK` message is received from a peer, the peer is mapped to a randomly selected CID observed during the simulation run, similar to what the first-spy estimator does. In addition to responding with a `FORWARD-HAVE`, the WFE still handles the request normally and either forwards it or starts a proxy session. Following this strategy, the WFE is able to correctly detect a high fraction of the nodes' interests without requiring knowledge of the privacy-subgraph.

The FSE and the WFE simply follow their strategies, and thus do not try to learn the privacy-subgraph and use information about the subgraph for detecting a peer's interest. This means the privacy-subgraph is unknown to these adversaries. According to the analysis in Section 4.2.7, a 2-regular subgraph should provide the best privacy in case of an unknown subgraph. In order to evaluate the impact of a known privacy-subgraph and provide results that can be contrasted to the WFE, a *Subgraph-Aware Want-Forward Exploiter* (SAWFE) is implemented. The SAWFE is aware of the topology of the privacy-subgraph and follows the strategy of the WFE with a simple heuristic added. In case of a known subgraph, the $\eta = \Delta(G)$ should provide the best privacy properties according to the analysis of Section 4.2.7. In the simulations, the SAWFE controls 10 nodes each of which connected to different 4 honest peers, exactly like the WFE. The SAWFE also maps every peer to the CID of the first `WANT-BLOCK` observed from this peer. After that, the WFE iterates over all observed `WANT-HAVE` messages, and for each assigns the CID to all unclassified direct subgraph predecessors of the node that sent this message. `WANT-HAVE` messages are only sent by proxies in RaWa-Bitswap, and thus the adversary can be sure that one preceding node is interested in the CID.

Since the higher the hop count the less likely the hop count occurs for a forwarded `WANT-FORWARD` message, it is most likely that one of the direct predecessors is interested in the CID. If a peer remains unclassified after these two strategies, the peer's interest is classified with an observed CID at random. The goal of introducing this adversary is only to show that a known subgraph provides worse privacy compared to an unknown one, and to explore the magnitude of worsening for the different subgraph variants. It does not aim to correctly classify as many peers as possible. Thus, a single simple heuristic is sufficient. The WFE and SAWFE are only run against RaWa-Bitswap since they are specifically designed for this protocol.

Simulated adversary	Assumption about privacy-subgraph	Strategy
First-Spy Estimator (FSE)	Unknown	Assign each peer to the first CID observed from it.
Want-Forward Exploiter (WFE)	Unknown	Respond to any observed `WANT-FORWARD` with a `FORWARD-HAVE` indicating a controlled node stores the block. Then, assign each peer to the CID of the first `WANT-BLOCK` observed from it.
Subgraph-Aware Want-Forward Exploiter (SAWFE)	Known	Follow the strategy of the WFE. Then, for each observed `WANT-HAVE`, assign all unclassified direct predecessors of the message's sender to the message's CID.

Table 5: Overview of the adversaries simulated in the Testground setup.

An overview of the simulated adversaries is presented in Table 5. The table does not include the adversaries' fallback strategy of mapping remaining peers to observed CIDs at random. The introduced forwarding in RaWa-Bitswap has the potential to influence not only the privacy but also the performance of the protocol [14]. The next section presents how this influence on performance is measured in the Testground setup.

5.1.4. Performance Metric

Performance of RaWa-Bitswap should be compared to Baseline Bitswap to understand the impact of the introduced forwarding mechanism. Similar to [14], the *time-to-first-block* (TTFB) is introduced as a metric of performance. TTFB represents the duration

from the user requesting the Bitswap implementation for content until the first block of this content (*i.e.*, the block of the root CID) is received. The TTFB is measured in seconds. This metric is reasonable due to RaWa-Bitswap's focus on content discovery. An alternative metric could be the time until a `FORWARD-HAVE` is received in response to a `WANT-FORWARD`. However, this metric is not considered since it would exclude the unforwarded search timer in the measurement, especially in cases when the requestor has to query the content routing subsystem itself. Additionally, this approach would be more challenging to compare with Baseline Bitswap, as Baseline Bitswap might send small blocks immediately in response to `WANT-HAVE` messages, as described in Section 4.4. When measuring performance, no adversarial nodes are simulated in the network.

Every content item in the simulations is 150 kiB in size and encapsulated in a single block. File sizes that would require multiple blocks are not considered, as only the TTFB is measured.

Since `WANT-FORWARD` messages are forwarded over multiple hops, it takes some time for a proxy to discover content providers. The proxy employs a similar approach to discovering providers as the session in Baseline Bitswap. A requestor in Baseline Bitswap can immediately execute this approach, whereas in RaWa-Bitswap, it takes time until the `WANT-FORWARD` reaches the node that decides to act as the proxy for the request. Therefore, the expected performance of RaWa-Bitswap is lower than that of Baseline Bitswap.

In the simulated scenario, nodes run their requests concurrently. As the nodes' interests are chosen at random from all available CIDs, it is possible for multiple nodes to be interested in the same CID. Consequently, one node might download the block before another. This situation provides peers with the option to download a block from more than one node, adding a form of replication to the testing setup. A slight degree of replication better resembles a real network situation [7].

5.2. Results

Multiple configurations applying different values of the protocol parameters to the simulation scenario are set up. Each configuration runs the scenario 100 times with the same parameter values to gather multiple samples of the metrics measured during the run. The measurements are visualized with boxplots in this section. In the boxplots, the boxes refer to the 25 % (Q1) and 75 % (Q3) quartiles of the sample set. The middle line represents the median of the data. The whiskers are drawn for the 1.5 interquartile range (IQR), displaying outliers outside of this range as + in the plot. Across all configurations, the Random Forward strategy is applied, and the parameters subgraph reconstruction timer, idle tick, and periodic search timer duration use the fixed values stated in Section 4.3. The configurations may measure privacy or performance of either RaWa-Bitswap or Baseline Bitswap. The configurations measuring privacy add an adversary to the network; the performance configurations are run without.

The different configurations aim to provide various types of results, which are explained in this section. Initially, the unforwarded search timer is evaluated to

determine the most useful duration for the subsequent simulation runs. Afterwards, the privacy and performance of RaWa-Bitswap are compared to Baseline Bitswap and to itself with different parameter values set. Finally, the vulnerability described in Section 5.1.3 is simulated, and the impact of a subgraph-aware adversary is measured.

5.2.1. Reasonable Unforwarded Search Timer

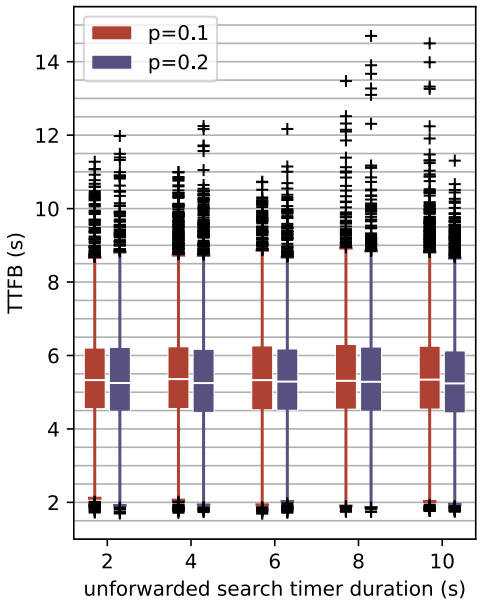

Figure 7: Performance of RaWa-Bitswap for various values of the unforwarded search timer duration.

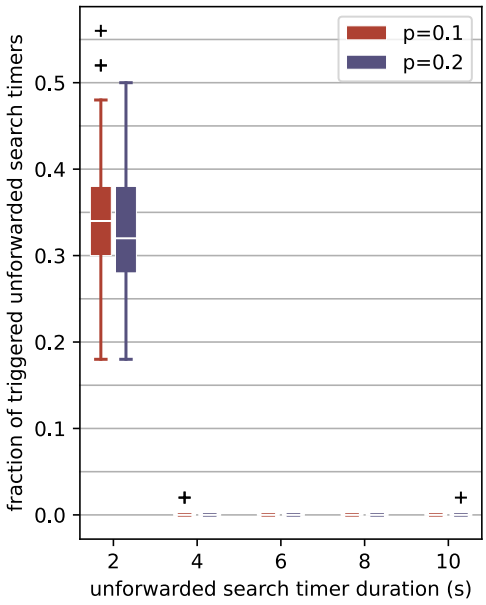

Figure 8: Fraction of nodes whose unforwarded search timer is triggered during a scenario run.

For evaluating the unforwarded search timer, the scenario is run with RaWa-Bitswap and varying values of the unforwarded search timer duration u. The goal is to find a value for u that demonstrates good protocol performance while limiting the frequency of unforwarded search timer triggers under the tested network configuration. This limitation is crucial because queries to the content routing subsystem by the requestor itself have the potential to reveal the node's interest to many peers, as discussed in Section 2.2, and these queries only serve as a fallback. The scenario is executed for 10 different combinations of the parameters u and p. For the unforwarded search timer duration, $u \in \{2, 4, 6, 8, 10\}$ is tested. Values of $p \in \{0.1, 0.2\}$ are used for the proxy transition probability. The variation of p is important as it determines the expected hop count for forwarded messages, thus influencing the likelihood of the timer triggering. The nodes approximate a 4-regular privacy-subgraph (*i.e.*, $\eta = 2$). Only the 4-regular graph is used in this setup since the choice of the privacy-subgraph does not influence the expected hop count. In this scenario, each of the 50 nodes queries one block,

setting the unforwarded search timer exactly once. The measurements include the Time-to-First-Block (TTFB) and the fraction of the 50 unforwarded search timers that trigger. Since the scenario is repeated 100 times, and each node downloads a block, the sample count for TTFB is 5000, and the sample count for the trigger fraction is 100 per configuration.

Figure 7 shows the boxplot of the TTFB for varying parameters. Across all different values for u, the TTFB is similar with a median of roughly 5.33 seconds for $p = 0.1$ and 5.26 seconds for $p = 0.2$. Hence, only between the configurations with different proxy transition probabilities does the TTFB show a tendency. The fraction of nodes whose unforwarded search timer is triggered is plotted in Figure 8. It can be seen that, for $u \geq 4$, only in outlying runs are any unforwarded search timers triggered. With $u = 2$, the $p = 0.1$ runs show a median of 34 % triggered timers, and the $p = 0.2$ runs show a median of 32 %. The frequency at which the unforwarded search timer triggers should be limited, but at the same time, u should be relatively small in order for the timer to quickly take action in exceptional cases. Since the performance did not differ significantly for the tested duration values, $u = 4$ is a sensible choice for the scenario and is used in all following configurations.

5.2.2. Baseline and Parameter Comparison

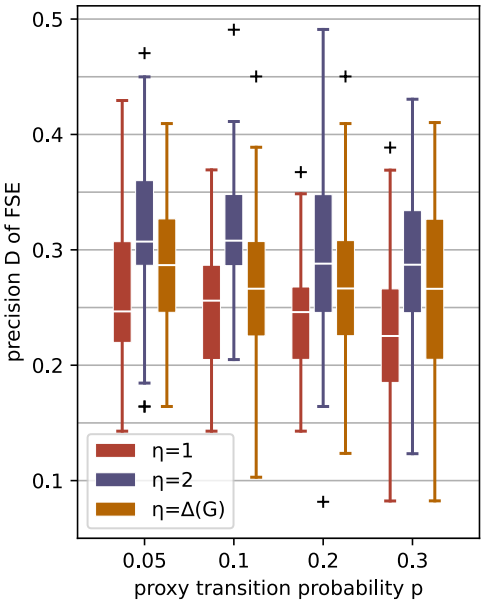

Figure 9: Precision of the First-Spy Estimator for RaWa-Bitswap.

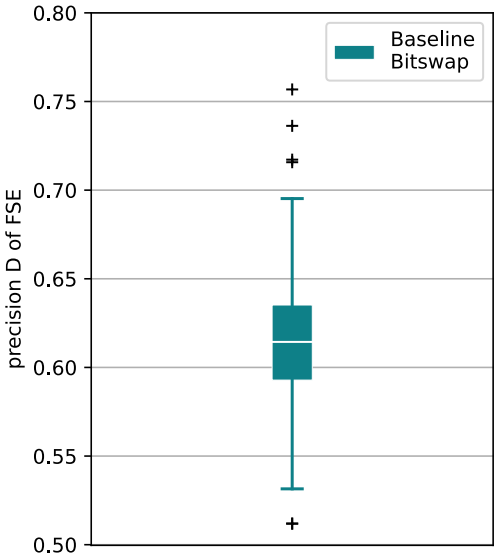

Figure 10: Precision of the First-Spy Estimator for Baseline Bitswap.

RaWa-Bitswap is compared to Baseline Bitswap in terms of privacy and performance. The comparison shows to what extent RaWa-Bitswap can provide better privacy than Baseline Bitswap when following the simple strategy of the FSE adversary. Since forwarding adds latency to content discovery, the performance changes are quantified by measuring the TTFB for both protocols. RaWa-Bitswap is run with various privacy-subgraphs and proxy transition probabilities. Values of $\eta \in \{1, 2, \Delta(G)\}$ and $p \in \{0.05, 0.1, 0.2, 0.3\}$ are configured. Varying these variables provides insight into their impact on privacy and performance. Baseline Bitswap is run with default settings. For the runs measuring privacy, each configuration introduces the FSE to the network. Thus, the number of honest nodes is 49. Each adversary produces one precision and one recall data point per run, resulting in 100 samples for each configuration. When measuring the TTFB, no adversary is added to the network, resulting in 5000 TTFB samples per performance configuration.

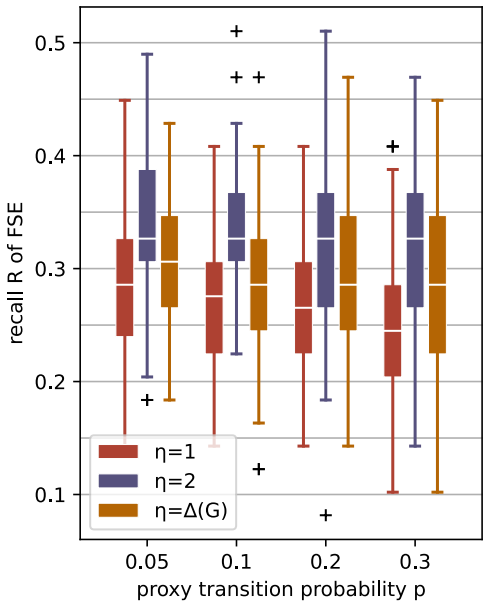

Figure 11: Recall of the First-Spy Estimator for RaWa-Bitswap.

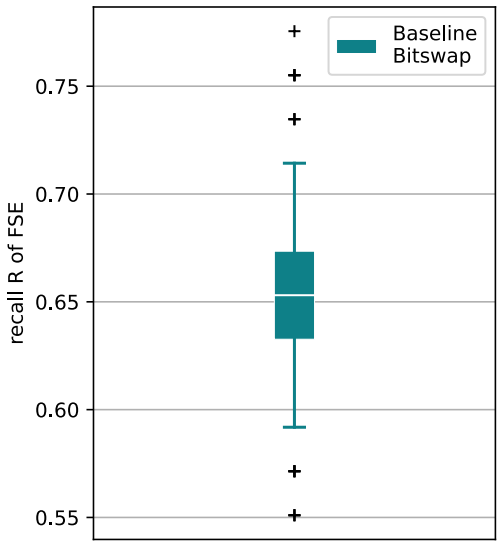

Figure 12: Recall of the First-Spy Estimator for Baseline Bitswap.

Figure 9 shows the precision of the FSE for RaWa-Bitswap, and Figure 10 for Baseline Bitswap. The precision of all runs with Baseline Bitswap is higher than the precision of all RaWa-Bitswap runs. The FSE recall of the RaWa-Bitswap runs can be seen in Figure 11, and of the Baseline Bitswap runs in Figure 12. Also, the recall of all runs with Baseline Bitswap is higher than the recall of all RaWa-Bitswap runs. Baseline Bitswap shows a high median precision of 0.61 and a high median recall of 0.65. This means the degree of plausible deniability per node is low, and the probability to correctly detect an

individual node's interest is about 65 %. In comparison, RaWa-Bitswap provides better plausible deniability and a lower probability to detect the interest of a node under the FSE. Both the median precision and median recall are two to three times higher with Baseline Bitswap than with RaWa-Bitswap under the FSE. The differences between the various parameter configurations of RaWa-Bitswap are similar for precision and recall. Among the different configurations of RaWa-Bitswap, the 2-regular subgraph with $p = 0.3$ shows the lowest medians for precision and recall. This configuration has a median precision of 0.23 and a median recall of 0.24. The worst median privacy under the FSE is shown by the 4-regular subgraph with $p \in \{0.05, 0.1\}$. Both configurations produce a median precision of 0.31 and a median recall of 0.33. When comparing the varying values of p for each of the subgraphs, no significant difference in privacy can be discerned. When comparing the varying subgraphs for each fixed p, the precision and recall for $\eta = 1$ are likely to be lower than for $\eta = \Delta(G)$. For $\eta = \Delta(G)$, these metrics are likely to be lower than for $\eta = 2$. This accounts for the 2-regular subgraph providing the best privacy under these circumstances.

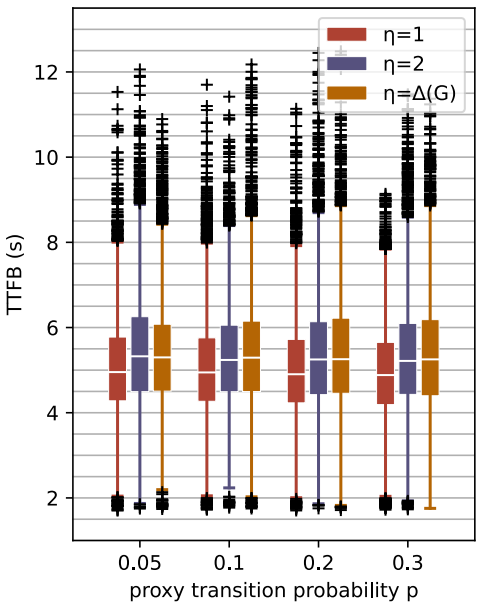

Figure 13: TTFB performance of RaWa-Bitswap.

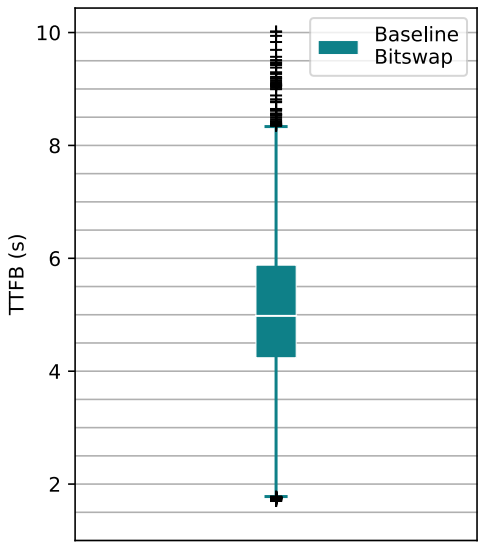

Figure 14: TTFB performance of Baseline Bitswap.

The measured performance of Baseline Bitswap and all configurations of RaWa-Bitswap are similar. RaWa-Bitswap's TTFB is plotted in Figure 13, and Baseline Bitswap's is shown in Figure 14. Baseline Bitswap's middle 50 % TTFB data points fall into the range of 4.24 to 5.88 seconds. For all $\eta \in \{2, \Delta(G)\}$ configurations, RaWa-Bitswap's Q1 and Q3 are respectively higher. For $\eta = 1$, the results are similar to Baseline Bitswap, yet $\eta = 1$ with $p = 0.3$ slightly exhibits the best performance

among all configurations. With this configuration, Q1 = 4.19, Q3 = 5.65, and a median of 4.88 seconds are measured for the TTFB. The highest median TTFB is recorded from the configuration of RaWa-Bitswap with $\eta = 2$ and $p = 0.05$, providing a median of 5.32 seconds. Considering the minimums and maximums without outliers, all configurations' data are highly spread. For example, Baseline Bitswap shows a minimum of 1.78 seconds and a maximum of 8.34 seconds. Similar to the privacy results under the FSE, no performance differences can be observed between varying proxy transition probabilities for each of RaWa-Bitswap's subgraph types, as the boxes are respectively plotted within similar ranges of TTFB values. When comparing the varying subgraphs, each single value of p shows a comparable range of the middle 50 % of TTFB measurements for $\eta \in \{2, \Delta(G)\}$. For $\eta = 1$, there is a tendency to provide lower middle 50 % values than the others. This indicates that the 2-regular subgraph tends to offer the best performance.

Only considering RaWa-Bitswap, both the best median privacy and the best median performance are shown for the 2-regular subgraph with $p = 0.3$. It is reasonable to assume that higher proxy transition probabilities tend to provide better performance because the higher the value of p, the fewer expected hops a forwarded message will have. Consequently, messages experience less delay when sent over network links. It is also sensible to assume that a higher p tends to result in better privacy under the FSE, as the FSE can only use the observed CIDs for classification. A higher p increases the likelihood that a forwarded message is proxied before the FSE can observe it, resulting in more interests being classified randomly. The assumption that higher proxy transition probabilities provide better privacy can be extended to the WFE and SAWFE adversaries, as they can only craft a bogus `FORWARD-HAVE` when a forwarded request reaches them. Consequently, only values of $p \in \{0.2, 0.3\}$ are configured in the following simulations, which are run with the WFE or the SAWFE included in the network.

5.2.3. Forwarding Vulnerability

As presented in Section 5.1.3, the RaWa-Bitswap protocol poses a vulnerability through the forwarding of messages when assuming that an adversary can craft arbitrary messages. To evaluate the impact of this vulnerability, simulations are conducted with the WFE introduced into the network. These simulations were performed for all subgraph types, but only for $p \in \{0.2, 0.3\}$, as reasoned in Section 5.2.2. The precision and recall of the WFE are measured for each run. These metrics are calculated from 40 node interests per run, in contrast to the FSE configurations which have 49 honest nodes, since the WFE controls 10 of the 50 nodes. Nonetheless, the sample count for precision and recall is 100 each due to the 100 simulation runs per configuration.

In Figure 15, the precision of the WFE is presented, and in Figure 16, the recall of the WFE is displayed. Compared to Baseline Bitswap under the FSE, RaWa-Bitswap exhibits lower precision across all configurations and runs under the WFE. Recall is also likely to be lower than in the Baseline Bitswap case. This implies a better plausible deniability and a lower probability of detecting an individual's interest

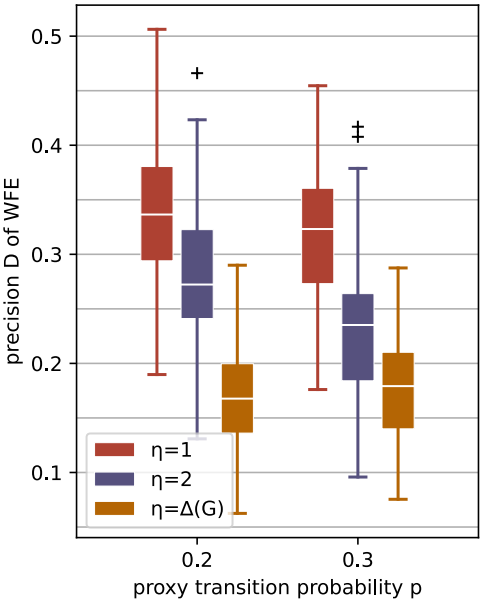

Figure 15: Precision of the Want-Forward Exploiter for RaWa-Bitswap.

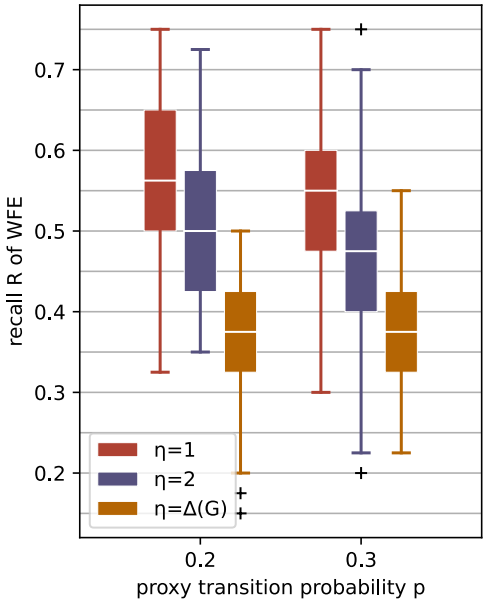

Figure 16: Recall of the Want-Forward Exploiter for RaWa-Bitswap.

for RaWa-Bitswap, even under a stronger adversary. In comparison to the privacy results of RaWa-Bitswap under the FSE, the precision remains relatively low across all configurations, while the gains in recall are significant under the WFE. Specifically, with $\eta = 1$, the precision of the WFE tends to be higher than that of the FSE. With $\eta = 2$ combined with $p = 0.2$, the precision of both adversaries is similar. However, with the combination of $\eta = 2$ and $p = 0.2$, and both configurations for $\eta = \Delta(G)$, the precision of the WFE tends to be lower. The recall of the WFE is consistently higher than that of the FSE for all configurations. This increase in recall is particularly pronounced for configurations that use a 2-regular subgraph. Under the WFE, the lowest medians for precision and recall are observed for $\eta = \Delta(G)$ with $p = 0.2$. This configuration has a median precision of 0.17 and a median recall of 0.38. The configuration with $\eta = 1$ and $p = 0.2$ exhibits the worst median privacy, with a median precision of 0.34 and a median recall of 0.56. Comparing the two values of p for the same subgraph, only the 4-regular graph shows a tendency to result in lower precision of the WFE with higher p. Apart from this, no significant difference in precision and recall is observed for varying proxy transition probabilities. For each value of p, the 2-regular subgraph tends to provide higher precision and recall than the 4-regular subgraph, and the 4-regular subgraph tends to offer higher precision and recall than the complete graph.

In conclusion, RaWa-Bitswap also demonstrates better privacy under the assumption

of the WFE than Baseline Bitswap. Overall, it maintains relatively good plausible deniability when this vulnerability is exploited, but the probability of detecting an individual node's interest is relatively high. The lowest median recall observed is 0.38, implying that an adversary can likely detect around 38 % of honest nodes' interests when following the WFE's strategy. Surprisingly, the 2-regular subgraph, which exhibited the best measured privacy properties under the FSE, performs the worst under the WFE. Since these two adversaries do not utilize any information about the subgraph's topology for classifying nodes' interests, the next section explores the precision and recall achievable when the subgraph is known.

5.2.4. Known Subgraph Privacy

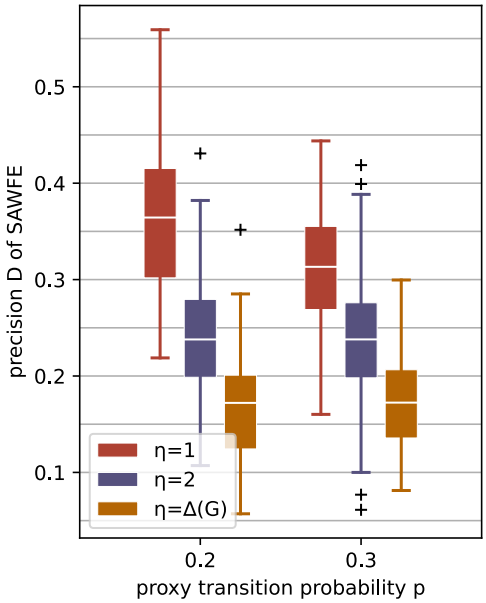

Figure 17: Precision of the Subgraph-Aware Want-Forward Exploiter for RaWa-Bitswap.

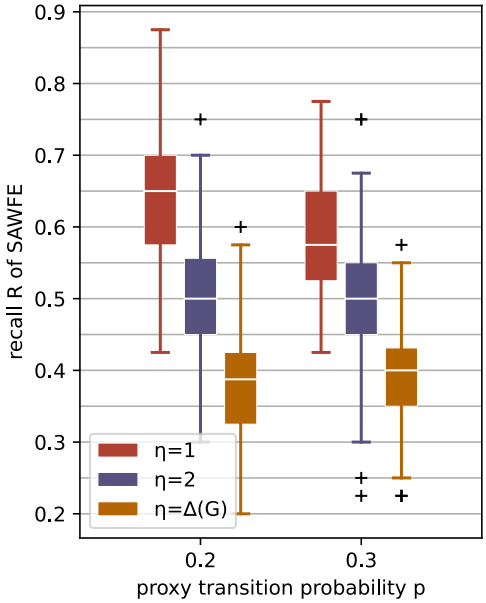

Figure 18: Recall of the Subgraph-Aware Want-Forward Exploiter for RaWa-Bitswap.

In this section, the changes in RaWa-Bitswap's privacy due to the adversary incorporating knowledge about the subgraph into the classification of nodes' interests are explored. To achieve this, simulations are conducted with the SAWFE adversary, which adopts the strategy of the WFE while incorporating a heuristic that utilizes knowledge of the subgraph. For comparison, the same set of parameter values as in Section 5.2.4 is employed. Precision and recall are calculated based on the interests of 40 honest nodes per run, with 100 samples per configuration.

The measured precision of the SAWFE is plotted in Figure 17, and its measured recall is shown in Figure 16. The precision results are comparable to those produced

by the WFE. Some parameter configurations exhibit higher median precision, while for others, it is lower, and no clear tendency can be observed, compared to the WFE. A slight trend is noticeable in the recall values. Compared to the same combinations of η and p under the WFE, the SAWFE's recall measurements, including Q1, Q3, and the median, are generally higher, except for the configuration $\eta = 2$ with $p = 0.2$, where the median is equal to and Q3 is slightly lower than under the WFE. The measurements also indicate a particularly high recall increase from the WFE to the SAWFE for the 2-regular subgraph configurations. When comparing the same combinations of η and p, the SAWFE exhibits higher or equal IQR for precision compared to the WFE, implying a more spread-out middle 50 % of precision data points. In contrast, the recall values show the opposite trend: under the SAWFE, the middle 50 % spread is lower or equal to that under the WFE for the same parameter configurations. Analyzing the results obtained by varying the parameters η and p, the changes in precision and recall between two configurations are similar to the WFE results. The best median privacy is achieved by $\eta = \Delta(G)$ with $p = 0.2$, demonstrating a median precision of 0.17 and a median recall of 0.39. The worst median privacy is reported for the configuration $\eta = 1$ with $p = 0.2$, yielding a median precision of 0.36 and a median recall of 0.65. Under the WFE, the highest and lowest medians for precision and recall are reported from the same parameter configurations. In summary, the measured changes in privacy are relatively minor compared to the WFE. However, there is a tendency for an adversary knowledgeable about the subgraph to have a higher recall, while the precision measurements remain similar in this experiment.

5.3. Discussion

The results of the conducted simulations fulfill the expected properties of RaWa-Bitswap in some aspects; however, in other aspects, they do not. In the following, a comparison to the expectations is explored, and the presented results are discussed under the assumptions posed by the testing setup. In particular, the impact of loops in the privacy-subgraph on the results under the tested network is analyzed. Furthermore, directions for future work are provided.

5.3.1. Comparison to Expectations

In the presented scenario, an ideal protocol would result in the highest recall if the adversary followed the strategy of assigning each node one CID at random. With this classification, a recall of $R = \frac{1}{|H|}$ is expected. However, since more than one node might be interested in a single CID, this strategy is not expected to show ideal precision. In the presented scenario, an idealistically low precision is achieved by an adversary that classifies each node with the same CID. The resulting precision depends on the number of peers that are actually interested in this CID. The highest precision under this classification strategy is achieved in the case that each node is indeed interested in this CID. Then, each peer is correctly classified (*i.e.*, $R = 1$) and the precision against each of the nodes is $D(h) = \frac{1}{|H|}$. Thus, this case results in a network-wide

precision of $D = \frac{1}{|H|}$. Under the assumption that the adversary also achieves at best the idealistic recall of $R = \frac{1}{|H|}$, the precision's bound is represented by the case where each node selects a different CID for downloading. In this case, only one peer is correctly classified, while the precision for each of the other nodes is 0. Therefore, the network-wide precision in this case is: $D = \frac{1}{|H|^2}$.

The ideal precision and recall provide insight into the extent to which RaWa-Bitswap improves privacy compared to Baseline Bitswap. However, the conducted measurements from RaWa-Bitswap under all adversaries are far from the ideal values. Overall, the lowest median recall is measured from the configuration $\eta = 1$ with $p = 0.3$ under the FSE. This configuration shows a median recall of 0.24, which is more than 10 times higher than the idealistic value $R = \frac{1}{|H|}$. Additionally, $\eta = \Delta(G)$ with $p = 0.2$ under the WFE and the SAWFE shows the lowest median precision of 0.17, which is even further away from the ideal value $D = \frac{1}{|H|^2}$. Nevertheless, the privacy gain compared to Baseline Bitswap is substantial. RaWa-Bitswap demonstrates better privacy under the stronger targeted WFE adversary than Baseline Bitswap under the weaker FSE. The FSE is able to correctly detect 65 % of the nodes' interests on average against Baseline Bitswap. This high recall is because a node asks every one of its neighbors for the content, thus revealing the node's interest. Since the FSE is connected to every node, it can observe all requests under Baseline Bitswap. The reason why the FSE's probability of detection is not 100 % is that the same CID might be requested several times, and the FSE only assigns newly observed CIDs deterministically.

The WFE adversary demonstrates that by forging `FORWARD-HAVE` messages, a high recall can easily be achieved against RaWa-Bitswap. It should be noted that the $\eta = \Delta(G)$ subgraph shows relatively low precision in the case of the forwarding vulnerability being exploited. The $\eta = \Delta(G)$ configurations even provide lower precision under the WFE than under the FSE, which accounts for better plausible deniability. Thus, even though the probability of detecting an individual node's interest is relatively high when this vulnerability is exploited, using the complete network graph as the privacy subgraph can still provide relatively good privacy from a network-wide perspective. Furthermore, this vulnerability only holds under the assumption that the adversary can craft arbitrary messages.

The expectations regarding the subgraphs, as implied by the analysis in Section 4.2.7, are only partially fulfilled by the results. In the case of a known subgraph, represented by the SAWFE adversary, the complete network graph indeed exhibits the best privacy. The situations involving an unknown subgraph are represented by the FSE and the WFE adversaries. Under the FSE, the 2-regular subgraph provides the lowest precision and recall as expected, but under the WFE, it does not. However, it's important to note that the aforementioned analysis is not directly applicable to the results under the WFE and SAWFE, as the analysis is based on Dandelion and Dandelion++. The forwarding vulnerability presented here is feasible because another request can be triggered by a response. In Dandelion and Dandelion++, no responses exist since they aim for transaction propagation. Nevertheless, the results indicate a tendency that the impact of the subgraph being known is highest in the case of the 2-regular subgraph.

No significant differences could be observed among the different values of p. The higher the value of p, the lower the expected hop count, and thus, better performance would be expected. Under ideal assumptions, precision increases with higher values of p, as demonstrated by the authors of Dandelion++ [18]. Nevertheless, a slight tendency of higher values of p reducing precision and recall is assumed for the tested conditions, as all three tested adversaries classify more interests at random when they do not observe a `WANT-FORWARD`.

In the tests conducted to determine a reasonable unforwarded search timer duration, it is unclear why the TTFB remains similar for all different values of u. The purpose of a triggered unforwarded search timer is to expedite the download process. Given that only with $u = 2$ a substantial fraction of timers trigger, one would expect to observe a change in TTFB, especially between $u = 2$ and $u = 4$. One possible explanation could be that when the timer triggers, the requestor needs to query the dummy DHT twice: once for the CID and once for the PeerID. This adds an average delay of 622 milliseconds twice before the provider can be requested. During this delay, a `FORWARD-HAVE` message can still be received. It is plausible that under $u = 2$, the delay is long enough for the requestor to download the block by utilizing a received `FORWARD-HAVE`, even though the timer has triggered. However, in the simulations, it is not recorded from which source each requestor acquires knowledge about the content provider. Therefore, this explanation cannot be verified.

Overall, the measured ranges of the TTFB are high, and the data is widely spread for both tested protocols. One reason for this observation could be that all peers are run on the same machine and all request content concurrently. This concurrency can overload the system, adding delays to the measured download times. Additionally, the dummy DHT introduces significant delays, which are likely to be added twice in most cases for both protocols: once for retrieving the PeerID of the provider and once for retrieving its multiaddresses. The intentional delay for each dummy DHT query provides an advantage in cases where a DHT query is unnecessary. For instance, if the multiaddresses of a provider can be obtained from a proxy, the requestor does not need to query the content routing subsystem, leading to faster block downloads. This setup also highlights the benefit of RaWa-Bitswap, where a DHT query can be saved when either the requestor or the proxy is connected to the content provider. However, it's crucial to note that the dummy DHT delay is derived from the public IPFS network, while other properties of the testing setup do not resemble real-world conditions. For instance, all nodes have the same bandwidth and link latency, and no nodes leave the network, omitting the evaluation of churn. Most notably, the tested network consists of only 50 nodes, which is much smaller than the IPFS network. Despite these limitations, the network setup provides valuable insights. It indicates a tendency that Baseline Bitswap's performance is better than RaWa-Bitswap's, except for the 2-regular graph configuration, which outperforms Baseline Bitswap.

The unexpected tendency of configurations with $\eta = 1$ showing lower TTFB measurements than Baseline Bitswap raises questions. One possible explanation is the difference in the expected number of content routing subsystem queries between Baseline Bitswap and RaWa-Bitswap. In Baseline Bitswap, the session has to query the

dummy DHT twice if no direct neighbor has the requested block. This is analogous to a situation in RaWa-Bitswap where no direct neighbor of the proxy has the block. In this scenario, the proxy requests the dummy DHT once for a CID and returns the found PeerID. Upon receiving the response, the requestor does not have to ask the DHT for the provider's multiaddresses if it is already connected to the provider. With the tested network of 50 nodes and 4 connections each, it is not unlikely that the requestor is already connected to the provider, saving one DHT query in RaWa-Bitswap. In RaWa-Bitswap, there are two opportunities to save the content routing subsystem query for the provider's multiaddress, whereas in Baseline Bitswap, there is only one chance. This property alone does not fully explain the performance of the 2-regular subgraph since other subgraph types show worse performance than Baseline Bitswap. However, this effect is amplified with the 2-regular subgraph because messages are proxied relatively earlier due to loops. The next section clarifies why and to what extent loops lower the expected hop count.

5.3.2. Impact of Loops

The expected number of hops for a proxied message is lower with $\eta = 1$ than the ideal expectation postulated by Equation 1 in Section 4.3. This discrepancy arises because the construction algorithm approximates a regular graph, which therefore might incorporate loops. This includes the case that two peers might choose each other as successors. When a peer receives a `WANT-FORWARD` message and cannot select a new successor for the CID, it immediately starts the proxy phase, as described in Section 4.2.5. This process notably impacts the results of the 2-regular subgraph, where each node has exactly one successor. Consequently, the proxy phase starts earlier for requests than expected, resulting in relatively fewer network links being passed per message. In the following section, an estimate is provided for the expected number of hops before encountering a loop in the approximated 2-regular subgraph.

It is assumed that each node's successor is uniformly chosen from all nodes in the network. As the path grows, each hop increases the likelihood that the next successor has been encountered before. Let N denote the set of all nodes in the network. The probability Q that the successor at hop e is encountered along the forwarded path before can be calculated using the following formula:

$$Q(e) = \frac{e}{|N| - 1} \tag{6}$$

To estimate the likelihood of encountering a loop, it is necessary to consider the probability to forward a message further on each hop (*i.e.*, $1 - p$). Therefore, the probability that hop e forwards the message into a loop is $(1-p)Q(e)$. The probability L_p that a message is forwarded into a loop within e hops under proxy transition probability p can be calculated from the complementary event:

$$L_p(e) = 1 - \prod_{i=1}^{e-1} 1 - (1-p)Q(i) \tag{7}$$

Equation 7 requires $e \geq 2$, since this is the first hop that can be a loop. By composing this formula with the probability that a message is proxied with e hops, as presented in Equation 1 in Section 4.3, the probability Y_p that a message is either proxied or forwarded into a loop within e hops is derived.

$$Y_p(e) = X_p(e)(1 - L_p(e)) + L_p(e) \tag{8}$$

The 2-regular subgraph shows, with $p = 0.3$, the lowest median TTFB. With $p = 0.3$ and the tested network size of $|N| = 50$, Equation 8 shows that a forwarded message is proxied or forwarded into a loop in at least 90 % of the cases within 6 hops, in at least 95 % of the cases within 8 hops, and in at least 99 % of the cases within 11 hops. The effect is even bigger for smaller p. In conclusion, these equations prove that the proxy phase starts, on average, earlier in the 2-regular subgraph than the ideal expectation due to the existence of loops. Notably, L_p reduces with a larger network, indicating that loops might have an impact on performance in the tested network, but this impact might be negligible in a real network.

The equations postulated in this section are only valid for a 2-regular privacy-subgraph. The situation is substantially more complicated with other subgraph types, as the probability to encounter a loop grows significantly with higher node degrees on each hop. Moreover, a single loop does not imply that a message cannot be forwarded further, as each node has more than one successor to choose from, with $\eta > 1$. Nevertheless, the estimation of the 2-regular subgraph case provides an impression of the order of the probability to encounter a loop for other subgraphs as well.

Loops might not only impact performance but also privacy. The results show that the 2-regular subgraph provides the best privacy properties under the FSE. The FSE can only use observed CIDs for classification, and thus classifies more interests at random when the FSE is reached by fewer messages. As it is expected that the proxy phase for messages is started significantly earlier than under ideal circumstances, it is also expected that the FSE encounters fewer messages. As discussed before, loops can influence the results of the 2-regular subgraph the most. The relatively high increase in precision and recall under the WFE and SAWFE, compared to the FSE, underscores these findings. Especially, the SAWFE is able to classify more peers correctly when the proxy phase is started early. With an earlier started proxy phase, it is expected that the SAWFE's heuristic is successful more frequently since, on average, more messages are proxied by the first hop.

The existence of loops can also explain the relatively low impact of varying p on privacy and performance since it influences the probability that the proxy phase is started early. The lower the value of p, the more likely it is to encounter a loop within

e hops.

The impact of loops could be resolved by choosing another successor as soon as a peer realizes that a received `WANT-FORWARD` cannot be forwarded further because all successors have already received it. This would resolve a loop and change the topology of the subgraph. Furthermore, in future work, it would be interesting to measure the number of hops after which a message is proxied in the simulations.

5.3.3. Future Work

Since the provided setup for evaluation focuses on identifying the privacy change compared to Baseline Bitswap, not all facets of RaWa-Bitswap are considered. This section provides an overview of aspects that future work should consider. Importantly, a direction for solving the WFE vulnerability in RaWa-Bitswap is presented.

The data exchange strategy considered in Section 4.2.2 is not utilized for the RaWa-Bitswap protocol due to a higher impact of churn, among other reasons. However, the extent of RaWa-Bitswap's robustness against the network's churn is not evaluated in the testing setup. Since higher churn decreases the likelihood that a `FORWARD-HAVE` message is successfully returned to the requestor, an important metric to capture in future work is the protocol's reliability under churn.

In Section 4.2.8, two strategies for selecting the successor which should receive a forwarded message are proposed. The presented testing setup only evaluates the use of the Random Forward strategy since it is likely to provide the best privacy properties for RaWa-Bitswap. Furthermore, the Closest PeerID Forward would require a specialized adversary for privacy evaluation. For performance evaluation, a different approach is necessary, as the dummy DHT would not mirror the performance gain of proxies closer to the CID. Since the privacy results of RaWa-Bitswap are not ideal, it is not sensible to introduce the Closest PeerID Forward, as it would worsen privacy.

The simulations for finding a reasonable duration of the unforwarded search timer used fewer variants of the parameters p and η than the other RaWa-Bitswap simulations. Variations of u are tested with a 4-regular subgraph and $p \in \{0.1, 0.2\}$. The resulting choice of $u = 4$ might not be applicable to parameter configurations that provide worse performance than the tested combinations of η and p. In Section 5.2.2, it is shown that the 2-regular subgraph tends to exhibit better performance than the others. Furthermore, the measured changes in performance for varying $p \in [0.05, 0.3]$ are negligible. Therefore, the choice of $u = 4$ is also applicable to the simulated configurations that are not explicitly tested in Section 5.2.1. However, these results are only applicable under the restricted conditions of the presented testing setup. Performance can differ in a real P2P network with many more nodes. For real situations, a sensible unforwarded search timer duration needs to be determined in future work.

The presented results do not answer the question of whether an adversary can learn the subgraph in a realistic scenario. The probability of this depends on the privacy-subgraph reconstruction timer, which is neglected in the evaluation. Using the presented testing setup, it could be experimentally evaluated how quickly a subgraph learning algorithm can be conducted. In Section 4.2.7, the subgraph learning algorithm

proposed by Sharma *et al.* is described as an example. However, the results would not be applicable to a realistic network, as the test network is relatively small with $|N| = 50$.

The WFE vulnerability is feasible because a `WANT-BLOCK` reveals the originator of the request, and a `WANT-BLOCK` can be triggered by a `FORWARD-HAVE`. At the same time, a message reveals, by design, the CID the originator is interested in, since it is not obfuscated. RaWa-Bitswap's assumption is that privacy can be achieved solely by obfuscating the source of a request. This assumption is sufficient in the case of Dandelion and Dandelion++. The protocols Dandelion and Dandelion++ do not involve a request-response model like Bitswap, as they are designed for transaction propagation in Bitcoin. Thus, obfuscating the source is enough to provide formal anonymity guarantees. To mitigate the WFE vulnerability in RaWa-Bitswap, it is necessary to additionally obfuscate the interests of the requestor. A recent study proposed three schemes for Bitswap, which achieve this obfuscation of interests [12]. Extending RaWa-Bitswap by one of these schemes might be a solution for mitigating the presented vulnerability. The schemes are discussed in detail in the next section.

6. Related Work

In this section, an overview of work related to RaWa-Bitswap is presented by exploring studies in the field of IPFS and Bitswap, as well as anonymous communication in general. IPFS has been extensively studied in various aspects, including the structure of its network [11, 23, 24], its performance [1, 58, 63, 65], its DHT [6, 25, 35], and security considerations [26, 37, 54, 60]. There are also publications that specifically address IPFS's underlying content exchange protocol, Bitswap. For example, De la Rocha *et al.* [14] provide an in-depth description of the Bitswap protocol, conduct performance measurements, and propose several protocol enhancements.

Several studies have focused on different aspects of privacy within the context of IPFS. In the work by Zhou *et al.* [66], they propose an encryption scheme that provides privacy for data stored in IPFS. Another study concerning data privacy in IPFS is the one by Lin and Zhang [29]. They employ a blockchain to store permission information for files, enabling access restrictions to data within IPFS. Balduf *et al.* [2] demonstrated that IPFS faces user privacy issues by introducing a methodology for monitoring Bitswap requests. Through the proposal and execution of various attacks, they showcased the practical feasibility of tracking users' interests and the vulnerability of users' privacy.

Two recent studies have addressed the privacy issues encountered by users who request content via Bitswap. Daniel *et al.* [9] introduced the concept of trickling into Bitswap to add plausible deniability and obfuscate the source of a request. In this approach, `WANT-HAVE` messages are forwarded through the network by sending them in rounds to the nodes' neighbors. Each new round starts after a fixed delay and targets a different group of neighbors. The source of a request can be predicted less accurately because an adversary may receive the message from multiple senders. Notably, RaWa-Bitswap differs from the trickling approach in terms of forwarding reach. Messages are not flooded through the network over multiple hops; instead, they are forwarded along a single path, resulting in RaWa-Bitswap imposing a lighter load on the network. In the other recent study [12], three different protocols for privacy-enhanced content discovery in Bitswap are proposed and evaluated: Bloom-Swap, PSI-Swap, and BEPSI-Swap. Bloom-Swap utilizes a Bloom filter to probabilistically discover the content a node stores, offering privacy to the requestor. However, this approach reduces provider privacy compared to Baseline Bitswap, as provider nodes reveal what content they store to some extent through the Bloom filter. Yet, nodes storing content also gain some plausible deniability because a Bloom filter is a probabilistic data structure. In PSI-Swap, both requestor and provider nodes can determine if a provider stores a specific block without revealing which block the requestor is interested in, using the cryptographic technique of private set intersection. The interest is only disclosed to the provider when sending a `WANT-BLOCK` message. While PSI-Swap introduces privacy in discovering content, it increases communication costs due to added computational overhead. In contrast, RaWa-Bitswap does not introduce computational overhead, as no additional cryptographic computations are incorporated into the messages. BEPSI-Swap combines elements of both Bloom-Swap and PSI-Swap by incorporating Bloom

filters into messages and performing the intersections on top of these. Thus, it inherits the advantages of both approaches, offering efficient privacy with providers gaining some plausible deniability. However, it still adds computational overhead. It is worth noting that, unlike RaWa-Bitswap, all approaches in [12] do not obfuscate the source of a request but obfuscate the interests of the requestor. Thus, extending RaWa-Bitswap by one of these approaches might solve the forwarding vulnerability presented in Section 5.1.3. Since PSI-Swap still reveals the interest when a `WANT-BLOCK` is sent, any of the two approaches based on Bloom filters seem more applicable. `WANT-FORWARD` and `FORWARD-HAVE` messages could relay Bloom filters, obfuscating the peers' interests. However, this would turn RaWa-Bitswap into a probabilistic protocol.

In RaWa-Bitswap, random walks are employed to obfuscate the source of a request and delegate content discovery to a proxy. Utilizing random walks for proxy selection has been previously proposed to enhance privacy in distributed systems. An early example is Crowds [56]. In this protocol, requests are forwarded over a random walk to a proxy, which executes a web transaction on behalf of another user. Crowds only provides anonymity guarantees at the level of individual users, while RaWa-Bitswap, in contrast, targets network-wide privacy. However, it has been demonstrated that Crowds achieves an optimal low value for the probability of detection when assuming a first-spy estimator [8]. When using stronger assumptions, such as more complex graph topologies and the objective of network-wide anonymity, Crowds falls short of optimality [5]. Consequently, Dandelion [5] and Dandelion++ [18] were proposed to address these stronger assumptions. By utilizing a privacy-subgraph for random-walk-based forwarding, Dandelion and Dandelion++ offer formal anonymity guarantees for the propagation of transactions in Bitcoin. Dandelion++ was implemented and deployed in the Monero blockchain [62]. In contrast to RaWa-Bitswap, the Dandelion and Dandelion++ publications do not evaluate the protocols' privacy in a realistic experimental setup, such as the Testground simulations conducted in this work. A simulated setup is also provided in the work by Sharma *et al.* [57]. They find that the anonymity of Dandelion and Dandelion++ is limited, although the assumptions of the three studies differ partially, as discussed in Section 4.2.7. An alternative approach to anonymous transaction propagation in Bitcoin is Clover [19], which is also based on random walks. While Clover does not require the construction of a subgraph, its authors demonstrated a similar level of anonymity compared to Dandelion++ in a simulation-based evaluation. The significant difference of RaWa-Bitswap compared to Clover, Dandelion, and Dandelion++ is that the RaWa-Bitswap protocol is able to route responses back along the same path as the request's random walk. Other notable P2P anonymity schemes utilizing random walks include Tarzan [20], AP3 [33], Rumor Riding [22], and ShadowWalker [34]. Compared to these, RaWa-Bitswap is the only protocol that utilizes a privacy-subgraph.

A prominent protocol for enabling private communication through proxies is Tor [16]. Requests are routed over a number of hops and sent by a proxy (*i.e.*, the exit relay) to a target host, while the communication is encrypted with session keys according to the Onion Routing scheme. Contrarily to RaWa-Bitswap, the path is chosen by the initiator and maintained for longer communication than sending a single message

and waiting for a response. Paths in Tor are also frequently changed to mitigate possible tracking attacks. Garlic Cast [55] is a protocol derived from the idea of Tor's underlying Onion Routing scheme and utilizes random walks to build paths for anonymous communication. McLachlan *et al.* [32] proposed Torsk, which is a privacy-enhanced approach for the selection of the relays in Tor. In this approach, any new relay performs a random walk for selecting buddies, which query a Kademlia DHT on behalf of the relay. However, Torsk is vulnerable to denial-of-service attacks [64].

7. Conclusion

In this study, source obfuscation through random-walk-based forwarding is explored to enhance the privacy of IPFS's content exchange protocol, Bitswap. Bitswap discovers content by initially querying all of its neighbors to determine if they store a CID-identified data item. Consequently, the CID being requested and the identity of the requestor are revealed to all neighbors. Previous studies have indicated that this information can be utilized to track user interests [2]. To address the privacy issues of Bitswap, this study proposes a protocol modification that obfuscates the originator of content discovery requests.

For the introduction of source obfuscation into Bitswap, this study analyzes Dandelion [5] and its advanced version, Dandelion++ [18]. These protocols were designed to introduce anonymity guarantees into Bitcoin's transaction propagation process. The concepts from Dandelion and Dandelion++ are extended in this study to create *RaWa-Bitswap*, a random-walk-based forwarding protocol aimed at enhancing privacy within a request-response model. In an experimental setup, RaWa-Bitswap is thoroughly evaluated and compared to Baseline Bitswap in terms of their privacy and performance properties. While the new protocol does not exhibit ideal characteristics, it significantly improves privacy for Bitswap users while maintaining performance under the tested conditions. Consequently, RaWa-Bitswap offers enhanced network-wide privacy compared to Baseline Bitswap. This improvement is evidenced by a reduced probability of detecting an individual node's interest and a higher degree of plausible deniability. RaWa-Bitswap incorporates various parameters. Their effects on changing privacy and performance are evaluated when varied. Among these parameters, the type of graph used for forwarding messages has the most significant impact on privacy and performance. Specifically, forwarding messages within the complete network graph, rather than utilizing a subgraph, results in the best-measured privacy under the most sophisticated adversary simulated. Importantly, this result has limitations, as the impact of forwarding loops varies for different graphs under the tested network, but this impact might be negligible under realistic conditions.

The RaWa-Bitswap protocol exhibits a vulnerability that limits the enhanced privacy properties of the protocol, when an adversary capable of crafting arbitrary messages is assumed. This vulnerability arises due to the disclosure of a request's originator through a `WANT-BLOCK` message and the absence of interest obfuscation. Therefore, RaWa-Bitswap's current privacy assumption, which focuses on obfuscating the request source, proves insufficient under stricter assumptions. To mitigate this vulnerability, RaWa-Bitswap could be combined with one of the recently proposed privacy-enhancing approaches introducing interest obfuscation into Bitswap [9]. Nevertheless, simulations indicate that, while an adversary has a relatively high probability of detecting an individual peer's interest when exploiting the vulnerability, RaWa-Bitswap still offers a relatively high degree of plausible deniability.

References

[1] O. Abdullah Lajam and T. Ahmed Helmy. Performance evaluation of ipfs in private networks. In *2021 4th International Conference on Data Storage and Data Engineering*, DSDE '21, pages 77–84, New York, NY, USA, 2021. Association for Computing Machinery.

[2] L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann. Monitoring data requests in decentralized data storage systems: A case study of ipfs. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 658–668, 2022.

[3] J. Benet. IPFS - content addressed, versioned, P2P file system. *arXiv:1407.3561v1*, 2014.

[4] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 15–29, New York, NY, USA, 2014. Association for Computing Machinery.

[5] S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath. Dandelion: Redesigning the bitcoin network for anonymity. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1), June 2017.

[6] L. Cao and Y. Zhang. Research on improvement of routing algorithm kademlia in ipfs. In *2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, pages 399–403, 2022.

[7] P. Á. Costa, J. Leitão, and Y. Psaras. Studying the workload of a fully decentralized web3 system: Ipfs. In M. Patiño-Martínez and J. Paulo, editors, *Distributed Applications and Interoperable Systems*, pages 20–36, Cham, 2023. Springer Nature Switzerland.

[8] G. Danezis, C. Diaz, E. Käsper, and C. Troncoso. The wisdom of crowds: Attacks and optimal constructions. In M. Backes and P. Ning, editors, *Computer Security – ESORICS 2009*, pages 406–423, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[9] E. Daniel, M. Ebert, and F. Tschorsch. Improving bitswap privacy with forwarding and source obfuscation. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, pages 1–4, 2023.

[10] E. Daniel and F. Tschorsch. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, 24(1):31–52, 2022.

[11] E. Daniel and F. Tschorsch. Passively measuring ipfs churn and network size. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, July 2022.

[12] E. Daniel and F. Tschorsch. Privacy-enhanced content discovery for bitswap. In *2023 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2023.

[13] A. de la Rocha. Github - adlrocha/go-bitswap - rfc|bb|l1-02: Ttls for rebroadcasting want messages. `https://github.com/adlrocha/go-bitswap/tree/feature/rfcBBL102`. Accessed: 2023-09.

[14] A. de la Rocha, D. Dias, and Y. Psaras. Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin. Technical report, Protocol Labs Research, 2021.

[15] B. Denby, A. Miller, G. Fanti, S. Bakshi, S. Bojja, and P. Viswanath. Dandelion - privacy enhancing routing. *Bitcoin Improvement Proposals*, 156, June 2017. `https://github.com/bitcoin/bips/blob/master/bip-0156.mediawiki`. Accessed: 2023-07.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, Aug. 2004. USENIX Association.

[17] M. Essaid, C. Lee, and H. Ju. Characterizing the bitcoin network topology with node-probe. *International Journal of Network Management*, 2023.

[18] G. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(2), June 2018.

[19] F. Franzoni and V. Daza. Clover: An anonymous transaction relay protocol for the bitcoin p2p network. *Peer-to-Peer Networking and Applications*, 15(1):290–303, 2022.

[20] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 193–206, New York, NY, USA, 2002. Association for Computing Machinery.

[21] Google LLC. Protocol buffers documentation - protocol buffers version 3 language specification. `https://protobuf.dev/reference/protobuf/proto3-spec/`. Accessed: 2023-09.

[22] J. Han and Y. Liu. Rumor riding: Anonymizing unstructured peer-to-peer systems. In *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 22–31, 2006.

[23] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann. Mapping the interplanetary filesystem. In *2020 IFIP Networking Conference (Networking)*, pages 289–297, 2020.

[24] S. A. Henningsen, S. Rust, M. Florian, and B. Scheuermann. Crawling the IPFS network. In *2020 IFIP Networking Conference, Networking 2020, Paris, France, June 22-26, 2020*, pages 679–680. IEEE, 2020.

[25] H. Kanemitsu, K. Kanai, and H. Nakazato. Lookup parameter optimization for kademlia dht alternative in ipfs. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 905–913, 2023.

[26] C. Karapapas, G. C. Polyzos, and C. Patsakis. What's inside a node? malicious ipfs nodes under the magnifying glass. *arXiv:2306.05541v1*, 2023.

[27] J. Kim and R. Srikant. Peer-to-peer streaming over dynamic random hamilton cycles. In *2012 Information Theory and Applications Workshop*, pages 415–419, 2012.

[28] P. Koshy, D. Koshy, and P. McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 469–485, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[29] Y. Lin and C. Zhang. A method for protecting private data in ipfs. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 404–409, 2021.

[30] L. Lovász. Random walks on graphs: A survey. In D. Miklós, V. T. Sós, and T. Szőnyi, editors, *Combinatorics, Paul Erdős is Eighty*, volume 2, pages 353–398. János Bolyai Mathematical Society, 1996.

[31] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Peer-to-Peer Systems*, pages 53–65, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[32] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 590—599, New York, NY, USA, 2009. Association for Computing Machinery.

[33] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. Ap3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, EW 11, New York, NY, USA, 2004. Association for Computing Machinery.

[34] P. Mittal and N. Borisov. Shadowwalker: Peer-to-peer anonymous communication using redundant structured topologies. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 161–172, New York, NY, USA, 2009. Association for Computing Machinery.

[35] J. Monteiro, P. Á. Costa, J. Leitão, A. De la Rocha, and Y. Psaras. Enriching kademlia by partitioning. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 33–38, 2022.

[36] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[37] C. Patsakis and F. Casino. Hydras and ipfs: a decentralised playground for malware. *International Journal of Information Security*, 18(6):787–799, 2019.

[38] Protocol Labs. Github - ipfs/boxo - boxo v0.8.0. `https://github.com/ipfs/boxo/tree/v0.8.0`. Accessed: 2023-09.

[39] Protocol Labs. Github - ipfs/boxo - boxo/bitswap/client/internal/messagequeue/messagequeue.go - defaultrebroadcastinterval. `https://github.com/ipfs/boxo/blob/v0.8.0/bitswap/client/internal/messagequeue/messagequeue.go#L25`. Accessed: 2023-09.

[40] Protocol Labs. Github - ipfs/boxo - go-bitswap. `https://github.com/ipfs/boxo/tree/v0.8.0/bitswap`. Accessed: 2023-09.

[41] Protocol Labs. Github - ipfs/kubo - kubo/config/init.go - default connection manager settings. `https://github.com/ipfs/kubo/blob/v0.20.0/config/init.go#L96-L102`. Accessed: 2023-06.

[42] Protocol Labs. Github - ipfs/specs - bitswap. `https://github.com/ipfs/specs/blob/main/BITSWAP.md`. Accessed: 2023-08.

[43] Protocol Labs. Github - ipfs/test-plans/bitswap-tuning - plan: Bitswap tuning - combinations of seeds and leeches. `https://github.com/ipfs/test-plans/tree/master/bitswap-tuning`. Accessed: 2023-09.

[44] Protocol Labs. Github - libp2p/specs - libp2p specification. `https://github.com/libp2p/specs`. Accessed: 2023-09.

[45] Protocol Labs. Github - libp2p/specs - peer ids and keys. `https://github.com/libp2p/specs/blob/master/peer-ids/peer-ids.md`. Accessed: 2023-07.

[46] Protocol Labs. Github - multiformats/cid - cid (content identifier) specification. `https://github.com/multiformats/cid`. Accessed: 2023-03.

[47] Protocol Labs. Github - multiformats/multiaddr - multiaddr. `https://github.com/multiformats/multiaddr`. Accessed: 2023-08.

[48] Protocol Labs. Github - multiformats/multihash - multihash. https://github.com/multiformats/multihash. Accessed: 2023-07.

[49] Protocol Labs. Ipfs docs - merkle directed acyclic graphs (dags). https://docs.ipfs.tech/concepts/merkle-dag/. Accessed: 2023-03.

[50] Protocol Labs. Ipfs docs - privacy and encryption. https://docs.ipfs.tech/concepts/privacy-and-encryption/. Accessed: 2023-06.

[51] Protocol Labs. Ipfs docs - what is ipfs? https://docs.ipfs.tech/concepts/what-is-ipfs/. Accessed: 2023-01.

[52] Protocol Labs. Ipld - dag-pb specification. https://ipld.io/specs/codecs/dag-pb/spec/. Accessed: 2023-03.

[53] Protocol Labs. Filecoin: A decentralized storage network. Technical report, 2017.

[54] B. Prünster, A. Marsalek, and T. Zefferer. Total eclipse of the heart – disrupting the InterPlanetary file system. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3735–3752, Boston, MA, Aug. 2022. USENIX Association.

[55] C. Qian, J. Shi, Z. Yu, Y. Yu, and S. Zhong. Garlic cast: Lightweight and decentralized anonymous content sharing. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 216–223, 2016.

[56] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66—-92, Nov. 1998.

[57] P. K. Sharma, D. Gosain, and C. Diaz. On the anonymity of peer-to-peer network anonymity schemes used by cryptocurrencies. In *Network and Distributed System Security (NDSS) Symposium 2023*, 2023.

[58] 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 '19, New York, NY, USA, 2019. Association for Computing Machinery.

[59] F. Spitzer. *Principles of Random Walk*. Graduate Texts in Mathematics, 34. Springer New York, NY, New York, NY, second edition, 1964.

[60] S. Sridhar, O. Ascigil, N. Keizer, F. Genon, S. Pierre, Y. Psaras, E. Rivière, and M. Król. Content censorship in the interplanetary file system. *arXiv:2307.12212v1*, 2023.

[61] Testground. Github - testground/testground. https://github.com/testground/testground. Accessed: 2023-02.

[62] The Monero Project. Github - monero-project/monero - pull request #6314 - adding dandelion++ support to public networks:. https://github.com/monero-project/monero/pull/6314. Accessed: 2023-09.

[63] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras. Design and evaluation of ipfs: A storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, pages 739–752, New York, NY, USA, 2022. Association for Computing Machinery.

[64] Q. Wang, P. Mittal, and N. Borisov. In search of an anonymous and secure lookup: Attacks on structured peer-to-peer anonymous communication systems. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 308–318, New York, NY, USA, 2010. Association for Computing Machinery.

[65] Z. Wu, C. R. Yang, S. Vargas, and A. Balasubramanian. Is ipfs ready for decentralized video streaming? In *Proceedings of the ACM Web Conference 2023*, WWW '23, pages 3002–3010, New York, NY, USA, 2023. Association for Computing Machinery.

[66] C. Zhou, G. Sun, X. You, and Y. Gu. A slice-based encryption scheme for ipfs. *International Journal of Security and Networks*, 18(1):42–51, 2023.

A. Message Specification

In the following, the specification of RaWa-Bitswap's message envelope is presented in the Protocol Buffers 3 format:

```
message Message {
  message Wantlist {
    enum WantType {
      Block = 0;
      Have = 1;
      Forward = 2;
    }

    message Entry {
      bytes block = 1;
       int32 priority = 2;
      bool cancel = 3;
      WantType wantType = 4;
      bool sendDontHave = 5;
    }

    repeated Entry entries = 1;
    bool full = 2;
  }

  message Block {
    bytes prefix = 1;
    bytes data = 2;
  }

  enum BlockPresenceType {
    Have = 0;
    DontHave = 1;
    ForwardHave = 2;
  }
  message BlockPresence {
    message AddressInfo {
      bytes peerId = 1;
      repeated bytes multiaddrs = 2;
    }

    bytes cid = 1;
    BlockPresenceType type = 2;
    repeated AddressInfo addrinfo = 3;
  }

  Wantlist wantlist = 1;
  repeated Block payload = 3;
  repeated BlockPresence blockPresences = 4;
   int32 pendingBytes = 5;
}
```

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 24. Oktober 2023 .....