Empirical and Analytical Perspectives on the Robustness of Blockchain-related Peer-to-Peer Networks

DISSERTATION
zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Sebastian Henningsen

Präsident (komm.) der Humboldt-Universität zu Berlin
Prof. Dr. Peter Frensch

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät
Prof. Dr. Elmar Kulke

1. Gutachter: Prof. Dr. Björn Scheuermann (HU Berlin)
2. Gutachter: Prof. Dr. Hannes Hartenstein (KIT)
3. Gutachter: Prof. Dr. Roger Wattenhofer (ETH Zürich)

Tag der Einreichung: 15.03.2021
Tag der mündlichen Prüfung: 9.03.2022

Empirical and Analytical Perspectives on the Robustness of Blockchain-related Peer-to-Peer Networks

Sebastian Henningsen

ABSTRACT

The inception of Bitcoin has sparked a large interest in decentralized systems, leading to significant research and development efforts being invested into building blockchain-based decentralized applications. These decentralized applications oftentimes mimic existing functionality and seek to replace their existing centralized counterparts without the alleged shortcomings. In particular, popular narratives imply that decentralization automatically leads to a high security and resilience against attacks, even against powerful adversaries.

In this thesis, we investigate whether these ascriptions are appropriate and if decentralized applications are as robust as they are made out to be. To this end, we exemplarily analyze three widely-used systems that function as building blocks for blockchain applications: Ethereum as basic infrastructure, IPFS for distributed storage and lastly "stablecoins" as tokens with a stable value. As reoccurring building blocks for decentralized applications these examples significantly determine the security and resilience of the overall application. Furthermore, focusing on these building blocks allows us to look past individual applications and focus on inherent systemic properties.

The analysis is driven by a strong empirical, mostly network-layer based perspective; enriched with an economic point of view in the context of monetary stabilization. The resulting practical understanding allows us to delve into the systems' inherent properties and transition beyond mere empirical considerations.

The fundamental results of this thesis include the demonstration of a network-layer Eclipse attack on the Ethereum overlay which can be leveraged to impede the delivery of transaction and blocks as well as to exclude honest participants from the network — with dire consequences for applications built on top of Ethereum. Furthermore, we extensively map the IPFS network through (1) systematic crawling of its DHT, as well as (2) monitoring content requests. We show that while IPFS' hybrid overlay structure renders it quite robust against attacks, this virtue of the overlay is simultaneously a curse, as it allows for extensive monitoring of participating peers and the data they request. Lastly, we exchange the network-layer perspective for a mostly economic one in the context of monetary stabilization. Deeply rooted in existing economic literature and concepts, we present a classification framework to (1) map out the stablecoin landscape and (2) provide means to pigeon-hole future system designs. With our work we not only scrutinize ascriptions attributed to decentral technologies; we also reached out to IPFS and Ethereum developers to discuss results and remedy potential attack vectors.

ZUSAMMENFASSUNG

Die Erfindung von Bitcoin hat ein großes Interesse an dezentralen Systemen geweckt, welches sich in einer Fülle an Forschungsarbeiten sowie der Entwicklung zahlreicher dezentralisierter Softwaresysteme niederschlägt. Diese dezentralisierten Anwendungen imitieren häufig bestehende zentralisierte, Systeme und versuchen die gleiche Funktionalität ohne die, angeblichen, Schwächen bereitzustellen. Eine häufige Zuschreibung an dezentrale Systeme ist dabei, dass eine Dezentralisierung automatisch zu einer höheren Sicherheit und Widerstandsfähigkeit gegenüber Angriffen führt.

Diese Dissertation widmet sich dieser Zuschreibung, indem untersucht wird, ob dezentralisierte Anwendungen tatsächlich so robust sind. Zu diesem Zweck werden exemplarisch drei Systeme untersucht, die häufig als Komponenten in komplexen Blockchain-Anwendungen benutzt werden: Ethereum als Infrastruktur, IPFS zur verteilten Datenspeicherung und schließlich "Stablecoins" als Tokens mit Wertstabilität. Die Sicherheit und Robustheit dieser einzelnen Komponenten bestimmt maßgeblich die Sicherheit des Gesamtsystems in dem sie verwendet werden; darüber hinaus erlaubt der Fokus auf Komponenten Schlussfolgerungen über individuelle Anwendungen hinaus.

Für die entsprechende Analyse bedient sich diese Arbeit einer empirisch motivierten, meist Netzwerklayer-basierten Perspektive — angereichert mit einer ökonomischen im Kontext von Wertstabilen Tokens. Dieses empirische Verständnis ermöglicht es Aussagen über die inhärenten Eigenschaften der studierten Systeme zu treffen.

Ein zentrales Ergebnis dieser Arbeit ist die Entdeckung und Demonstration einer "Eclipse-Attack" auf das Ethereum Overlay. Mittels eines solchen Angriffs kann ein Angreifer die Verbreitung von Transaktionen und Blöcken behindern und Netzwerkteilnehmer aus dem Overlay ausschlieSSen. Des weiteren wird das IPFS-Netzwerk umfassend analysiert und kartografiert mithilfe (1) systematischer Crawls der DHT sowie (2) des Mitschneidens von Anfragenachrichten für Daten. Erkenntlich wird hierbei, dass die hybride Overlay-Struktur von IPFS Segen und Fluch zugleich ist: Auf der einen Seite ist das Gesamtsystem robust gegen Angriffe, auf der anderen Seite ist eine umfassende Überwachung der Netzwerkteilnehmer möglich. Im Rahmen der wertstabilen Kryptowährungen wird ein Klassifikations-Framework vorgestellt und auf aktuelle Entwicklungen im Gebiet der "Stablecoins" angewandt. Mit diesem Framework wird somit (1) der aktuelle Zustand der Stablecoin-Landschaft sortiert und (2) ein Mittel zur Verfügung gestellt, um auch zukünftige Designs einzuordnen und zu verstehen.

ACKNOWLEDGEMENTS

This thesis concludes a long research journey with many people, changes in direction and influences along the way. However, it is a journey with too many people to name them all, so please do not feel discouraged if you do not appear in this expression of gratitude – chances are high I still hold your influence dearly.

First, I would like to express my gratitude to my advisor Björn Scheuermann, who employed me all these years and has enabled countless opportunities for personal growth. Furthermore, I want to thank my two reviewers Roger Wattenhofer and Hannes Hartenstein.

My scientific journey began with my bachelor's thesis in Kaiserslautern with Jens Schmitt and Michael Beck, who sparked my interest in computer network research. Without their continuous dedication, patience and support I would not be where I am today. Jens and Michael, I cannot thank you enough.

During my time in Björn Scheuermann's group, I've had the pleasure to meet and work with many inspiring people, the exchange with whom has improved my skills and work significantly. Among many others, Florian Tschorsch, who has always been a role model for me (which he was most likely not aware of), as well as Stefan Dietzel. Stefan, thank you for your numerous and extensive comments on my early papers, which has really helped me to structure my written thoughts and make them more accessible. In that period, I shared my office with Roman Naumann and Samuel Brack, thank you both for the numerous discussions and support on work-related problems, your perspectives have helped me in my research countless times. Besides work, I also had a great time slacking in 4321 with you.

The majority of papers for this thesis was written at the Weizenbaum Institute, a remarkable place which has broadened my horizon dramatically, but with too many people that I hold dear and enlightening connections to name them all. Therefore, I will focus on my research group: Martin Florian, Ingolf Pernice, Roman Proskalovich, Sophie Beaucamp, Moritz Becker and Leonhard Balduf.

Martin, this thesis would not have been possible without you. Your insights, comments, questions and (moral) support were what enabled me to finish this dissertation, thank you so much! Ingolf, spending time and working with you has always been a pleasure, stay as humble and empathetic as you are. Similarly for Roman, thank you both for the enlightenment in economics and thank you Roman for introducing me to the wonderful confectionery Zefir. Sophie, while your analytical skills never cease to impress me, I am mainly fascinated by your ability to match my cheekiness, which has made working with you

very enjoyable. Thank you for insights into the intricacies of the law and for exploring various places in Berlin with me. Moritz, thank you for introducing me to the critical and meta perspective of social science which makes me question everything. Leo, thanks for reintroducing me to the concept of clean code, after years of academic coding. Last but not least, I want to thank our assistant Jana Pinheiro for the good vibes and the immense load of trouble that she shielded away from us.

During my PhD I have supervised a total of 26 bachelor's and master's theses and had the pleasure to work with many talented students, most notably Leo Balduf, Ansgar Lößer, Sebastian Rust, Daniel Teunis and Joel Witzke. Especially Daniel, Leo, and Sebastian have contributed significantly to the evaluation sections of this thesis, thank you all!

Last but not least, my friends and family contributed significantly to the success of this dissertation through their support and encouragement in difficult times, by enduring my stressful phases and simply by providing safe spaces for me. Glenn Daly, Fabian Kalt, Marei Kerschl, Marlene Metzger and Kamila de Pasquale, thank you for the wonderful time in Berlin. Simon Birnbach, Thomas Lottermann, Lukas Übelacker, Frederik Walk and Sebastian Wolff, thank you for the endless hours in discord and for friendships that defy vast spatial distances. I would also like to express my gratitude towards my parents, Doris Fischer-Henningsen and Michael Henningsen, who gave me the room and possibilities to thrive while inspiring me to follow in their steps as a researcher. Thank you all for everything, I could not have wished for better people in my life.

My deepest gratitude deserves my girlfriend, Marie Sofie Jacob. Marie, you continue to fascinate and inspire me every day. I cannot thank you enough for the endless support and love, positive energy and freedom that you provide me.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION & PROBLEM STATEMENT

The advent of Bitcoin [202] and the accompanying hype in "blockchain technology" has sparked a renewed and significant interest in decentralized systems without a central authority. Since Bitcoin's inception, a plethora of alternative cryptocurrencies (so-called altcoins) have seen the light of day [9, 257], increasing the interest in "blockchain"[1] to such degrees that it has reached even government actors [3, 259].

This hype in blockchain is driven by a popular story: the "trustlessness"[2] of Bitcoin and other cryptocurrencies, i.e., the ability to reach consensus and, as a consequence, conduct payments, among distrusting actors without a trusted third party, liberates people from (allegedly) failure-prone intermediaries, such as banks. A common inference and narrative from the alleged inherent decentralization of such systems is their robustness against (Denial-of-Service-) attacks and censorship — even in the presence of powerful adversaries like state actors.

But are practically deployed applications as decentralized and robust as they are made out to be? Are these ascriptions appropriate?

In this thesis we tackle these questions by studying specific systems, which are widely adopted in the real world, from a conceptual and empirical perspective. Particularly, we argue and showcase that decentralization does not automatically entail robustness.

Blockchain applications are employed in a variety of different contexts, examples include but are not limited to:

- decentral autonomous organizations (DAOs), decentralized systems for governing social communities [74],
- self-sovereign identities [82],
- decentralized finance (DeFi), partially automated and complex financial derivatives [268], and
- distributed file-hosting [154].

Since Bitcoin's inception in 2009 as a decentralized payment system, blockchain applications have evolved dramatically into highly complex systems involving multiple stand-alone components, whose composition is more than the sum of the individual parts. In this thesis, we refrain from studying specific and entire applications, as the degree of complexity is prohibitive and, additionally, any such assessment would be outdated quickly. Instead, we disentangle this complexity

[1] *With "blockchain" we refer to open, permissionless, systems without fixed identities, e.g., Bitcoin, and not Byzantine agreement systems.*

[2] *Other disciplines have criticized computer science's narrow definition of "trust" as trust in, for example, Bitcoin is merely shifted towards developers and exchanges [31].*

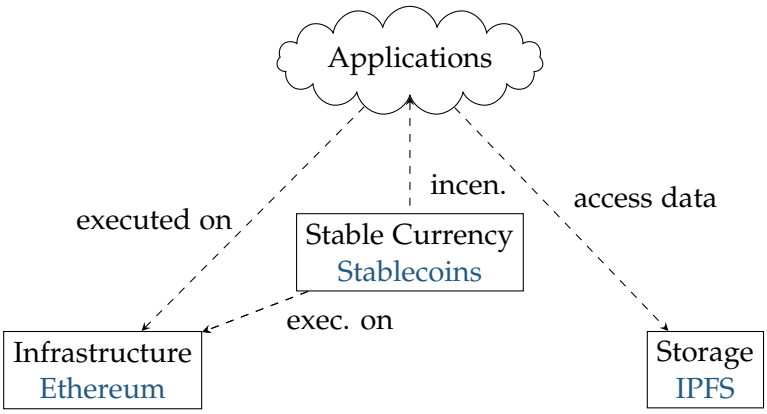


Figure 1: Blockchain application stack as well as the respective systems that we will investigate in the course of this thesis.

by focusing on individual components from different layers of the "blockchain application stack" model (introduced in detail in Section 2.6); the idea being that the robustness and decentralization of a blockchain application as a whole are only as good as application's weakest component.

We identified three important building blocks of blockchain applications[3], which are gathered in a blockchain application stack model[4] depicted in Figure 1 — all of which will be covered throughout this thesis. The infrastructure block is inspired by the insight that Bitcoin, Ethereum and the like are not only used for decentralized payments, but are rather seen as an infrastructural basis for building more complex applications on top of them. These dApps rely on properties and guarantees provided by the underlying blockchain system in terms of consensus and lack of intermediaries in order to build novel, distributed and decentralized applications. Having a blockchain infrastructure in place is not sufficient for many applications, however, as storing larger quantities of data directly "on-chain" is prohibitively expensive. Therefore, dApps use other peer-to-peer (P2P) systems as a storage layer for storing their application data, which are then cryptographically linked to the decentralized execution on some kind of infrastructure. Lastly, due to the inherent lack of a central entity for rule-enforcement in decentralized systems, participants are instead motivated to fulfill relevant tasks for the system through (monetary) incentives. One such example is the mining of new blocks in permissionless blockchains in, e.g., Bitcoin, where participation in the process yields monetary rewards in the form of bitcoins (cf. Section 2.3). However, the value of these coins fluctuates dramatically, potentially leading to misaligned incentives. Therefore, we argue that monetary stabilization is a crucial building block for blockchain applications for maintaining stable incentives.

In this dissertation, we devote one chapter to each layer, commencing with the infrastructure layer in Chapter 3, the storage layer in

[3] *Oftentimes also referred to as decentralized applications (dApps).*

[4] *Loosely inspired by non-scientific ideas of a decentralized application stack, e.g., [95],*

Chapter 4 and closing with the layer of monetary stabilization in Chapter 5. From each layer, we will study one representative system which serves as an illustrative example of layer-inherent properties. The representative example systems from each layer that we will study are: Ethereum at the infrastructural layer, the Interplanetary Filesystem (IPFS) as a popular solution for distributed storage and stablecoins, as an essential building block for monetary stabilization. To study their robustness we will mostly adopt a network layer perspective (in correspondence to the OSI model), i.e., focus on gaining a thorough understanding of the overlay to assess possible attack vectors on the network layer as well as their implications for the system as a whole. Therefore, the research focus lies mostly on the P2P networks and overlays on which blockchain- and other decentralized systems are based upon. Towards the end of this thesis, when studying monetary stabilization, we will instead adopt a combination of economic and technical perspectives. Specifically, we will investigate economic robustness, i.e., studying the implications of adverse market conditions and rational economic behavior on the stability of stablecoin systems in relation to their technical implementations.

This thesis is primarily based on previously published collaborative work [1, 6–9]. Despite the focus on gaining an empirical understanding, the aim of these works (and therefore this thesis) is to transition beyond empirical snapshots and instead gain insights into system-inherent properties. Hence, the goal is not to give a cutting-edge, up-to-date description of Ethereum, IPFS and stablecoins, especially since the blockchain ecosystem is evolving at a rapid rate, but rather gain a deeper understanding.

The attentive reader might wonder whether the example systems on each layer, Ethereum, IPFS and stablecoins, are reasonable representatives. We would argue so, as prominent dApps utilize at least one representative system:

- Aragon [74], one of the largest DAO platforms, is mainly based on Ethereum and IPFS,
- Microsoft ION [82], a large-scale platform for decentralized, self-sovereign identities, builds on top of Bitcoin and IPFS,
- Filecoin [154] and BitTorrentToken [39] combine a cryptocurrency to incentivize distributed file hosting on IPFS and software forks thereof, and
- decentralized finance (DeFi) [126, 222] uses stablecoins to build complex financial derivatives.

1.2 OUTLINE & CONTRIBUTIONS

Commencing on the infrastructure layer, Ethereum is the second largest cryptocurrency after Bitcoin (cf. Section 2.3) by market capitalization and the first to enable the creation of smart contracts — small programs that are executed redundantly by every node in the network. Due to these smart contracts, Ethereum is popular among application developers and acts as the infrastructural basis for a variety of projects [74, 126, 270]. In Chapter 3 we show that Ethereum is vulnerable to eclipse attacks, i.e., attacks on the network layer that enable Denial-of-Service (DoS), double spending of funds, advantages in the mining process and endanger the availability of decentralized applications. Our attack targets the Kademlia-based peer discovery of Ethereum — an ill-suited design choice as Kademlia is not optimized for building overlays that can withstand deliberate attacks (cf. Sections 2.2 and 2.4.2). Instead of overwriting the whole Kademlia-based discovery table with a multitude of Sybil nodes as in a previous attack by Marcus et al. [168], we specifically target the interplay between peer discovery and connection management. In particular, we exploit the public structure of Ethereum's peer discovery logic by inserting a small number of carefully crafted Sybil identities which are favored over other benign nodes during connection establishment. Therefore, despite the subnet restrictions implemented in Go Ethereum (Geth) v1.8.0 in response to the preceding attack by Marcus et al. [168], we only need *two* IP addresses from distinct /24 subnets for a successful attack.

During the process of responsible disclosure with the Ethereum foundation[5] and the resulting personal exchange with their developers, we designed hot fixes to address our attack (implemented in Geth v1.9.0 and subsequent versions). Although mitigating the immediate threat, we argue that these countermeasures are not enough: Ethereum should rather abstain from Kademlia altogether and instead move towards a more suitable peer discovery logic (e.g., Bitcoin's Addrman, cf. Section 2.4). Thus, given our attack, the narrative of decentralization entailing robustness is challenged.

[5] *Yielding rank 13 (at the time of writing) within the top 20 of Ethereum's bug bounty leaderboard (bounty-.ethereum.org/).*

Moving on to the storage layer, several systems are emerging concurrently, all of them seeking to establish a storage layer for decentralized applications, such that the data does not have to be stored directly "on" the blockchain, which would be costly. Examples of storage solutions include IPFS [34, 6], BitTorrentFS [39] and Swarm [106] — for a qualitative comparison, the reader is referred to [70]. In this thesis, we focus on Interplanetary Filesystem (IPFS) as the representative of the storage layer (cf. Chapter 4), as it is the most mature and furthest-developed distributed storage system. The question arises whether IPFS is actually suited to fulfill this role in terms of robustness and

decentralization. Especially censorship and DoS are viable threats that a distributed storage solution has to cope with.

To this end, we analyzed IPFS' inner workings and monitored the IPFS network from several perspectives. IPFS resembles classical P2P networks, particularly filesharing systems, in that content and peers are managed through a combination of a Kademlia-inspired distributed hash table (DHT) and flooding to one-hop neighbors. This peculiar combination safeguards IPFS from suffering the same fate as Ethereum in Chapter 3 through a similar attack. For our monitoring study, we exploit both the Kademlia-based DHT lookups as well as the one-hop flooding of content requests:

- by crawling[6] the Kademlia-based DHT to obtain a (partial) view of the overlay graph,
- by running monitoring nodes which were connected to a large share of the network, and
- by sniffing the one-hop flooding traffic of content requests, yielding a trace of *who* requested *what* data *when*.

[6] *The respective crawler is maintained at github.com/wiberlin/-ipfs-crawler.*

Equipped with these perspectives we give a thorough view of IPFS' overlay and content-distribution network. We conclude that the network is still relatively decentralized in that most traffic is generated by normal users and not infrastructure nodes. Furthermore, through sacrificing performance and user privacy, IPFS achieves resilience against Sybil and network partitioning attacks. However, with a similar setup to ours, an adversary can easily extract which nodes host specific content, which can be leveraged for DoS attacks on blockchain applications storing their data on IPFS.

Last but not least, we shift the focus to monetary stabilization in cryptocurrencies, implemented in the form of so-called "stablecoins" in Chapter 5. Stablecoins promise the best of two worlds: a permissionless cryptocurrency like Bitcoin, combined with the price stability of fiat currencies (e.g., USD, Euro) [9]. This combination makes them an interesting target to research, as their goal lies in the realm of economics, while their implementation is purely technical. Asking about the robustness and resilience of these coins is therefore not only bound to technical considerations as in the two preceding chapters, but also related to stability under adverse market conditions and rational agents. For this endeavor we present a comprehensive taxonomy on monetary stabilization in cryptocurrencies, taking account insights from monetary theory of traditional fiat currencies from existing economic literature as well as specificities of cryptocurrencies. The application of this taxonomy yields a classification and deep understanding of the stablecoin landscape, allowing us to reason about fundamental properties and limitations instead of individual projects. Equipped with our framework, we argue that many stablecoins are

either use problematic combinations of stabilization techniques and stability goals which could render them vulnerable against economic attacks, or not fully permissionless, due to the involvement of trusted parties. Since our publication, significant work with a similar hybrid perspective has been conducted on the general stability of decentralized finance (DeFi) [126, 145, 146, 222, 281].

In summary, we provide strong empirical perspectives on real-world systems which we leverage to gain in-depth insights on inherent systemic properties and limitations on the portrayed systems itself. We argue that decentralization does not automatically entail robustness. On the contrary, decentralized systems have to be carefully designed to withstand a variety of potential attack vectors while maintaining their decentralization. Furthermore, we bridge the gap between scientific insights and practice through collaboration with the respective developers; reflected, e.g., in the changes in Go Ethereum v1.9.0 and IPFS. Before diving into the technical details of the eclipse attack on Ethereum in Chapter 3, we first give an introduction to the P2P paradigm, permissionless blockchains and the role of P2P overlay structure in these blockchain systems.

2 PRELIMINARIES

2.1 A MOTIVATION TO PEER-TO-PEER-NETWORKING

Resembling "blockchain", which is en vogue at the time of writing, peer-to-peer (P2P) networks have been a subject of interest as well in the 1990s and 2000s — attracting significant research and development effort, resulting in an abundance of research papers, implementations and products. Due to the immensity of material, any introduction to P2P networking is necessarily incomplete. Therefore, we will focus on a concise overview of ideas and developments, especially changes in narratives, that are important for this particular thesis. For further introduction, the reader is kindly referred to the excellent books [44, 165, 247] and surveys [14, 260, 262] on the topic.

First, what are P2P networks? In this thesis we adopt the following definition:

Definition 1. *The P2P paradigm is the antagonist to classical client-server architectures in that there is no clear distinction between nodes serving requests of client nodes but that each node can act as client and server simultaneously.*

Other definitions [165] follow the literate meaning of "peer", furthermore requiring an egalitarian system, i.e., in which peers are equally privileged and equipped with equal powers. However, this addition excludes hybrid architectures (e.g., Gnutella [248, 251, 274], early Skype [28, 127]) which differ from the traditional client-server model and can thus be considered P2P systems as well. Hence, we argue that egalitarian systems such as the ones covered in this thesis (e.g., Bitcoin, Ethereum, IPFS) are merely a sub-form of the general P2P paradigm.

In most P2P networks, participants establish application layer connections over the Internet, forming a so-called *overlay*. Overlay proximity is, therefore, logical and does not necessarily correlate with geographic proximity, i.e., nodes can be logical neighbors while residing on different continents. The shape and properties of the resulting overlay graph have severe impacts on the robustness against failures and attacks as well as the performance of the overall network. The use case dictates the requirements, e.g., low latency, high resilience against failures, etc., and an overlay must be designed accordingly. Hence, the design of a P2P system (and it's resulting overlay) necessarily involves trade-offs and, as we will see in Chapter 3, choosing an overlay unsuitable for the specific use case can even impose security risks. Overlays can be roughly divided into three non-exclusive categories:

unstructured, hybrid with central servers and structured. [44, 247]. As a rule of thumb: that the more structure there is, the better the performance (e.g., latency of content retrieval) of the network at the cost of attack resilience.

Initially, the main narrative around the benefits of P2P networks was performance[7] (e.g., [152] and [247, p. 9]): with growing interest in digital technologies and more widespread adoption of the Internet, centralized servers were not able to scale, especially when transferring large files, with bandwidth being a major bottleneck. Kumar and Ross [152] elegantly modeled P2P file distribution and showed that when peers provide their bandwidth to distribute data themselves, the system can scale to an unlimited number of users.

[7] *Since there are no systematic studies, there can only be hints to support the claim, e.g., searching the ACM digital library for "peer-to-peer" AND "scalability in abstract" yields 2560 results, whereas "censorship" only yields 83.*

The distribution of large data items to a set of recipients was subsequently the subject of optimization, leading to a variety of structured overlays, especially in the form of distributed hash tables (DHTs). The key idea behind these systems is a typical divide & conquer approach: assign keys to data items in the form of hashes and partition the key space, such that each peer is responsible for only a subset of the key space (and corresponding data items). Among the many proposals of such DHTs, *Kademlia* [170] and variants thereof are the most frequently used in practice, e.g., in BitTorrent [217], eMule [244], IPFS [6], Ethereum [8], I2P [278]. There are several conceivable reasons for Kademlia's popularity in practice, as will be elaborated on in Section 2.2. In summary, it displays a logarithmic (in the total number of peers) lookup performance with low overhead in terms of connection management while being easy to implement and robust against churn. While other approaches may be superior in theory, Kademlia seems to achieve a good balance between implementation difficulty and performance in the real world.

The narratives of better performance and scalability of P2P systems in comparison to client-server architectures were challenged by advances in other areas, as highlighted in Mager et al. [164] by the example of the Wuala file distribution service. Wuala, an online storage service similar to Dropbox [87], deliberately transitioned from a hybrid P2P architecture in which peers distributed data among themselves together with a central server towards a classic client-server model. In an interview Luzius Meisser [164], the co-founder of Wuala, states several reasons for this decision: (1) a price-drop of about one order of magnitude in bandwidth costs, (2) marginal contribution of peers to data distribution, (3) complexity of hybrid architectures in comparison to a central-server one and (4) prevalence of mobile and battery constrained devices.

The significant lower bandwidth costs diminished the economic incentive which, apparently, was the main driving factor to endure the additional complexity of a hybrid architecture. To fully reap the benefits of peer-assisted data distribution and storage, one has

to maintain a healthy overlay in the face of churn and unreliable peers, which involves additional communication overhead [41, 227]. In the presence of an ever-increasing number of battery constrained devices [164], additional communication (which equals additional energy consumption) is highly undesirable.

Naturally, Wuala is only one example application, but we argue that the four arguments stated for a switch towards a centralized architecture instead of a P2P one are applicable to the P2P paradigm in general, when performance and scalability are the main selling points. Incidentally, Wuala's transition came at a time when interest in P2P networks was declining in general[8], as can be seen in Figure 2, which depicts the Google n-gram "popularity" of the term P2P over time. The Google n-gram dataset consists of a large collection of books; hence, the y-axis of Figure 2 depicts the relative frequency of the terms "Peer-to-Peer network" and "p2p network"(case insensitive). It can be seen that the relative frequency of P2P reached a peak around 2005 with a steady decline until 2015, at which point it rises again—potentially due to the increasing interest in Bitcoin and its overlay which commenced at that time [131]. However, the declining interest in the topic does not imply

[8] *The discontinuation of the IEEE conference on Peer-to-Peer Computing or the USENIX workshop on Peer-to-Peer Systems serve as additional hints.*

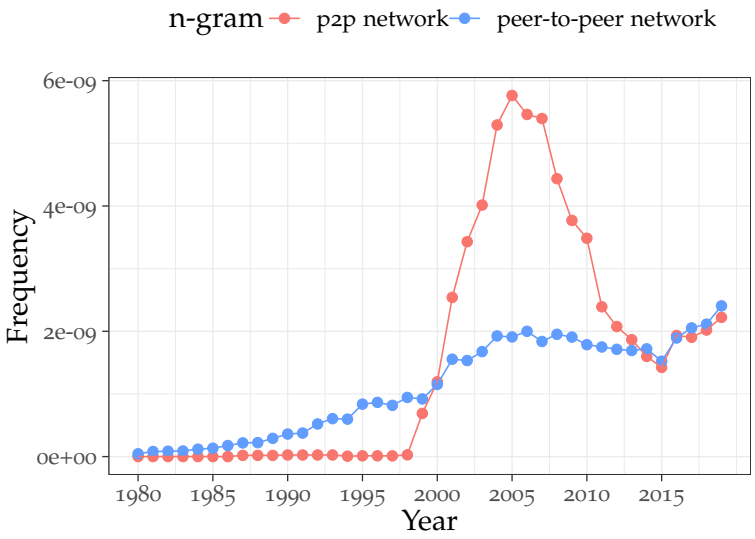


Figure 2: The frequency of P2P n-grams over time as a proxy for interest in the topic (obtained using Google's n-gram search).

that the P2P paradigm in general is not useful. In contrast, some ideas of P2P networks were incorporated in other technologies (e.g., [71] and [247, p. 13f.]), whereas other systems with different design goals than performance/scalability prevailed. The TOR network [254, 256], Bitcoin [202] and related cryptocurrencies [256], BitTorrent [217], IPFS [34, 6] as well as Freenet [229] are united by shared narratives: Instead of performance and scalability, these P2P systems focus on resilience against attacks (and/or law enforcement in general) and

censorship-resistance due to an inherent lack of central authority. This change in perspective is also reflected in the overlay structure, as can be illustrated by the projects that are covered in this thesis: Bitcoin (cf. Section 2.4) uses an unstructured overlay with flooding — robust but not efficient. Ethereum (cf. Chapter 3) combines flooding with Kademlia, a non-optimal combination which renders Ethereum vulnerable to eclipse attacks, as will be demonstrated in the corresponding section. IPFS combines Kademlia-based lookups with one-hop flooding (e.g., flooding to all direct neighbors) to retrieve content, making it robust against Sybil and related attacks (cf. Section 2.4.1) but deliberately sacrificing performance and user privacy for this additional robustness.

In the following we dig deeper into Kademlia, a DHT which underlies Ethereum as well as IPFS, before introducing Bitcoin and Ethereum.

2.2 KADEMLIA

Both Ethereum and IPFS implement their own variants and flavors of Kademlia which differ from the presented textbook version in some aspects. In the respective chapters (Chapter 3 and Chapter 4) we will highlight the implications of these differences when diving deeper into the matter.

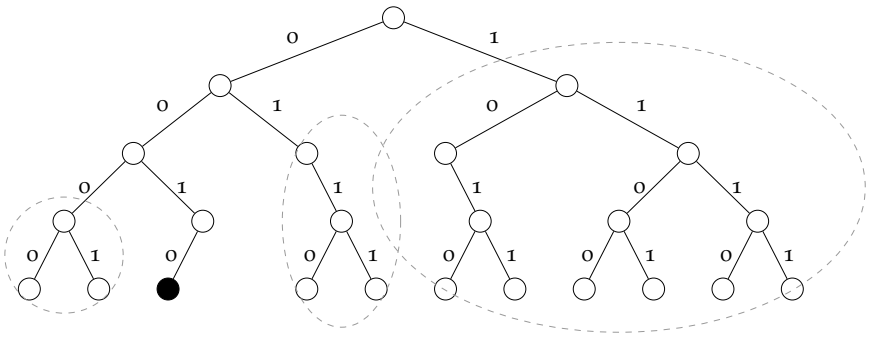


Figure 3: Kademlia structures its overlay according to the xor distance between nodes. When pictured as a binary tree, the marked buckets arise for the black node.

Kademlia is a UDP-based, P2P DHT that is used to locate decentrally stored data efficiently [170]. Similar to other DHTs, Kademlia imposes a structure on the participating peers, as well as the data they store, and provides a *lookup* mechanism to iteratively approach a target ID in a logarithmic number of steps. To that end, each node is uniquely identified by its randomly generated node ID — in Ethereum and IPFS the 256-bit[9] SHA-2 hash of a public key [34, 271]. Data items share the same representation: a data item's key is the 256-bit hash of the data itself.

[9] *Node IDs and keys in the original Kademlia proposal are 160-bit long.*

Node and data IDs are effectively treated as the leaves of a binary tree, as depicted in Figure 3. The leftmost leaf of the tree corresponds to the node with ID $0000\ldots$, or 0^b, $b = 256$ in regular language representation. Similarly, the rightmost leaf holds the node ID $1111\ldots$, 1^b. Each nodes structures the ID space by partitioning the binary tree into subtrees in dependence on the distance to its own node ID. Consider node N with ID(N) = 0011 ($b = 4$ in this case), as shown in Figure 3, as an illustrative example. The *first* subtree contains all nodes whose ID differs from ID(N) = 0011 in the *first* bit, i.e., every ID starting with 1: $\{1000, 1001, \ldots 1111\}$. Similarly, the second subtree contains all nodes whose ID differs in the second bit, yielding $\{0100, 0101, 0110, 0111\}$. Therefore, nodes in the i-th subtree differ in the i-th bit, thus, the last subtree only contains ID(N) itself. Put differently, the first subtree is "responsible" for half of the ID space, the second for $\frac{1}{4}$, etc.

The partition into subtrees depends on the length of the common prefix between two node IDs, or, equivalently, on the number of leading zeros of their XOR. The equivalent representation in terms of XOR is important with regards to Kademlia's definition of (logical) closeness: the closeness of two IDs x, y is defined as their bitwise XOR, taken as an integer value, i.e., $d(x, y) := x \oplus y$. The xor metric is symmetric, i.e., $\forall x, y : d(x, y) = d(y, x)$, yielding a simpler, yet still effective, routing structure than, for example, in Chord [170]. Due to this symmetry, nodes can elegantly update their routing tables while being traversed during another node's lookup. Given the xor distance metric, subtree i contains those node IDs whose distance to ID(N) is in $[2^i, 2^{i+1})$. Hence, the smaller the subtree number, the closer the corresponding node IDs are to N in terms of d: the first subtrees contain node IDs that are logically far away, whereas later subtrees's IDs are logically close.

In practice, nodes do not store entire subtrees (as the first subtree would contain about half of the network) but rather subsets of these trees of size k. These subsets are called k-*buckets* (or simply bucket) and store up to k neighbors[10]. Due to the size limitations, nodes "know" all IDs in their (logical) proximity while having some shortcuts to (logical) distant nodes. Furthermore, when bootstrapping, a node N performs a lookup for its own ID ID(N), to which other nodes return their closest neighbors to ID(N). Due to the symmetry of $d(.,.)$, these are also the closest neighbors to N, effectively populating the higher-index buckets which hold the nodes in close logical proximity to N. Bucket entries are implicitly sorted through a "least-recently-seen" policy, implicitly implementing a bias towards long-running nodes.

[10] *In IPFS* $k = 20$, *in Ethereum* $k = 16$.

Intuitively, this explains the logarithmic number of steps to lookup data: in Kademlia, data is stored at nodes whose node ID is "close" to the data's key. During a lookup, nodes iteratively[11] query their closest neighbors to a target ID, effectively traversing at least one subtree in

[11] *In the original Kademlia proposal the lookup is handled recursively.*

each lookup step until the desired content is found. In summary, in a network with N peers, Kademlia achieves a lookup performance of $O(\log_b(N))$ while maintaining $O(b \cdot \log_b(N))$ routing table entries at each node [247].

Lookups in Kademlia are effective when nodes collaborate and follow the protocol and experience crash failures at most. However, things change in light of deliberate adversarial behavior: similar to other DHTs, Kademlia is vulnerable to a variety of attacks due to the deterministic lookup [29, 115]. Most importantly, Kademlia is susceptible to Sybil (and related) attacks, which we will cover in Section 2.4.1 and Section 2.5.

2.3 INTRODUCING BITCOIN & ETHEREUM

Bitcoin, proposed by Satoshi Nakamoto[1] in 2008 [202] on the Cryptography mailing list [203] is the first fully decentralized digital cash system, also called a "cryptocurrency". In that, Bitcoin ends the 25-year long search for a fully decentralized digital cash system without trusted third parties, by cleverly combining existing ideas and techniques such as linked hash/signature chains [129, 169], Hashcash Proof-of-Work [20], b-money [65], Merkle Trees [177] and Byzantine Quorum systems [155, 166]. This combination yields a linked hash chain of so-called *blocks*, where each block's hash depends on the preceding block while additionally being a Proof-of-Work (PoW). Due to the interlinkage of blocks the resulting data structure is commonly referred to as *blockchain*[12]. Since its advent, Bitcoin has started a massive hype around the theme of "blockchain" which has since become a nebulous term covering various concepts, ranging from distributed databases to actual cryptocurrencies [272]. Furthermore, said hype has lead to a plethora of alternative cryptocurrency designs, a vast body of scientific research [19, 26, 43, 125, 257], government plans on building up "blockchain infrastructures" [259] and a variety of startups in the financial and banking sector [135].

[12] *Originally "block chain", the term has since evolved into a separate noun.*

Ironically, the inception of Bitcoin is closely related to the financial crisis in 2007/2008 and is rooted in the libertarian cypherpunk and crypto-anarchistic movements [252] — seeking to replace the existing banking system rather than fostering it [142].

The entanglement with the financial crisis is even written in Bitcoin's so-called *genesis block*, i.e., the first block of the blockchain which was created by Satoshi Nakamoto themselves[13]. Not only does this first block represent the transition of Bitcoin from a mere proposal into an actual running system, it is also hard-coded into the Bitcoin source

[13] *General term, since the gender and number of persons behind the pseudonym are unclear.*

1 The name is a pseudonym and it remains unknown who Satoshi Nakamoto is. While it is conjectured the pseudonym stands for an entire group, several individual suspects have been discussed. Despite the lack of definite proof on Satoshi's identity/identities, there exists one good heuristic on the matter, namely, that anyone *claiming* to be Satoshi Nakamoto is not the real person.

code to bootstrap new peers joining the network. The genesis block contains the message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks", referring to a headline in "The London Times" on the U.K. considering a second bailout for struggling banks in the midst of the financial crisis in 2008/2009.

In contrast to the traditional banking system, in the original Bitcoin proposal there is no place for trusted third parties such as banks and nation states [252]. The combination of these crypto-anarchistic and libertarian roots and the time of financial instability may, on the one hand, explain Bitcoin's initial appeal to a variety of people and the resulting hype around "blockchain". On the other hand, another reason for Bitcoin's success could lie in the fact that Bitcoin and related systems are academically pleasant to study: Firstly, Bitcoin's individual parts are surprisingly simple to grasp; it is their combination and interplay that constitutes a complex, decentralized P2P digital cash system. Secondly, Bitcoin solves a computer science problem that has remained open for 25 years, opening up plenty of new ideas to study, use cases to explore and systems to design.

One of these new systems is Ethereum [271], the meanwhile second largest cryptocurrency after Bitcoin by market capitalization. Ethereum enhances Bitcoin's design by enabling users to not only send payments but conditioning payments on arbitrarily complex logics. Whereas Bitcoin's main goal is a payment system, Ethereum leverages the same principles to build a decentralized, heavily redundant computing infrastructure, accompanied by a payment system. Ethereum is the subject of study of Chapter 3 and partially of Chapter 5 in this thesis. Therefore, in the following, we give a technical introduction to Bitcoin and Ethereum, the two largest cryptocurrencies at the time of writing. Although Bitcoin is not the explicit subject of study in this thesis, understanding the necessary basics of Ethereum is significantly easier when explained from the vantage point of Bitcoin. Hence, the technical introduction starts with a brief description of Bitcoin (cf. Section 2.3.1) and transitions into Ethereum later on (cf. Section 2.3.2). Note that this introduction is necessarily incomplete. For more details, the reader is advised to consult the Bitcoin and Ethereum white papers [202, 271]. Furthermore, Tschorsch and Scheuermann [257] contains an exhaustive description of Bitcoin and paints a bigger picture of Bitcoin's place in academic literature; as does Wattenhofer [265] for Byzantine Agreement Systems.

2.3.1 *Bitcoin Primer*

2.3.1.1 *Transactions*

In a nutshell, Bitcoin is a technology that enables the possibility to achieve a consensus among non-trusting participants in an open sys-

tem without previously established identities, also called a permissionless system. Permissionless in the sense that anyone can join or depart the system at any time and there exists no instance that assigns identities to entities. This lies in stark contrast to traditional Byzantine consensus and quorum approaches which require a-priori knowledge of participating entities [265, 272]. Intuitively, Byzantine consensus algorithms establish a consensus even in the presence of Byzantine adversaries, i.e., arbitrary node faults and attacks, through robust voting; thus in an permissionless environment these systems would be highly vulnerable to Sybil attacks (cf. Section 2.4.1).

Bitcoin solves the consensus problem in permissionless settings by lifting the consistency-assumption of classical Byzantine quorum systems: consensus in Bitcoin is only *eventually* consistent and there is no computable function to determine the point in time when it will be consistent [265]. In other words, while traditional Byzantine fault tolerance (BFT) protocols ensure one consistent view on the shared data of all network participants whenever an update to the shared is issued, Bitcoin only gives the guarantee of the existence of a point in time where all views will be consistent. Hence, in theory, whenever funds are moved through Bitcoin, the participating parties have no mechanism to compute whether their exchange is part of the consensus and can be considered final. In reality, Bitcoin overcomes this theoretical limitation through clever heuristics which yield points in time when participants can be sufficiently sure that their exchange of funds is part of the consensus.

The consensus in Bitcoin is established over a ledger containing coin ownership information, i.e., the amount of coins each "account" holds. This ledger is distributed and replicated at every node through a P2P network, hence, without updates to the ledger, each node has the same state[14].

[14] *In that, Bitcoin is more a heavily replicated state machine than a database, a commonly used term to explain the technically involved system to non-technicians.*

Consider the example in Figure 4 of Alice and Bob — an example which will re-occur throughout this section. Alice and Bob both own certain amounts of Bitcoin, n in the case of Alice and m in the case of Bob. It has to be noted that coin ownership in Bitcoin differs from ownership in traditional banking in that coins are not stored in accounts associated to identities but rather "in" asymmetric key pairs, which are called *addresses*. Alice owns a set of coins, identified by a hashed public key, if she can prove that she owns the corresponding private key. Therefore, in Bitcoin's design and narrative, ownership is defined solely through knowledge of secret keys — everybody with the private key to an address is, for the Bitcoin system, a legitimate owner of the funds in that address[15]. Another difference is that coins are not enumerated as in the real world or in Chaum et al. [52, 53].

[15] *This notion of ownership is barely relatable to the legal definition of everyday life. In this, this one vivid example of "code is law" [157, p. 4ff.].*

For the sake of explanation, we assume that Alice has exactly 5 coins in one address. To send 5 coins to Bob she creates a so-called *transaction* T with her address as input and Bob's address as output; though more

Transaction T
By: Alice
Unlock A_1
Lock in B_2

A_1: 5.0 BTC

A_2: 2.3 BTC

A_3: 0.1 BTC

B_1: 3.0 BTC

B_2: 0.0 BTC

Figure 4: A transaction of 5 BTC from Alice to Bob. Icon attribution: [133, 134].

inputs and outputs are possible in general. The transaction T contains a signature of the transaction data using Alice's private key (thereby also ensuring the transaction's integrity). This unlocks *all* five Bitcoins in input A_1 which must be spent completely by assigning them to new outputs (in this case B_2), as all unspent coins are collected as fees by miners. Subsequently, she broadcasts this transaction to all her neighboring peers which spread the information to all other nodes in the network through flooding. All receiving peers check the validity of the transaction, e.g., if the signature matches the public key, so that it is only forwarded if it passes all validation checks. Eventually, Alice's transaction has been spread throughout the overlay network to every peer and has entered each peers *memory pool (mempool)*: the set of all transactions which have been recently observed but are not yet part of the consensus.

2.3.1.2 *Blocks & Consensus*

The consensus is formed through a process called *mining*. To that end, a subset of peers in the network participates in the mining process (these peers are also referred to as *miners*) in that they take a set of transactions and gather them into so-called *blocks* in exchange for a monetary reward in the form of newly created Bitcoins and transaction fees. Blocks contain meta data and a list of transactions. In general, transactions are only considered valid if they are included in a block and said block is, subsequently, appended to the current blockchain. But a block cannot simply be created; instead a computationally intense Proof-of-Work has to be solved in order to create a valid block. In its design, Bitcoin intends to resemble gold in the real world, solving the PoW, which creates new Bitcoins in the process, parallels the mining for gold — hence the name.

Intuitively, blocks fulfill two roles:

1. establishing a total order among the transactions, and

2. restricting "write-access" to the shared ledger, or, equivalently, restricting the rate of state transitions of the Bitcoin system as a whole.

The total ordering is used to avoid inconsistencies in the state of the shared ledger. Without a total ordering of transactions different peers could potentially end up with different ledgers — inconsistencies which would render Bitcoin inoperable. Regarding the second role, without limits on block-creation, Sybil attacks would be possible (cf. Section 2.4.1) and data races between simultaneously-issued blocks would be common[16]. Requiring the solution to a PoW with every block creation limits the rate at which blocks are found and therefore restricts the rate at which the shared ledger is advanced through transactions.

[16] *Even with a PoW data races still occur in Bitcoin, but a lot less frequent (cf. Section 2.3.1.4).*

Figure 5 depicts the structure of a block. A block consists of a block header (light gray) containing meta information and a block body (white) containing the transactions. The block header comprises of

- a SHA-256 hash over the entirety of the block header,
- the hash of the previous block,
- a `nonce`[17] (for the mining process),
- the current time (with some margin for error), and
- the root hash of the Merkle-tree in the block body, in which the transactions are stored as leaves; thus yielding a total ordering through the hash tree structure.

[17] ***N**umber used **once**.*

Finding a block involves altering the `nonce` and `time` fields of the block header, such that the resulting SHA-256 hash of the block starts with a pre-defined number of zeros. Due a hash functions' collision and pre-image-resistant nature the PoW is hard to compute but easy to verify. Therefore, mining involves brute-forcing SHA-256 hashes until a suitable combination of `nonce` and `time` is found whose resulting block hash exhibits enough leading zeros. Verification is as simple as computing the hash of the block header (in which a block's transactions are included through the Merkle-root of the transaction tree) and checking, if the resulting hash of the block header starts with the pre-defined number of zeros. The effect of each block pointing to one predecessor can be seen in Figure 5: it yields an append-only structure of chained blocks[18], thus the name *blockchain*. Due to the structure of a hashed list, data inside the blockchain is immutable, since block B_i's hash depends on the block hash of B_{i-1} and any change of information in block would result in new block hashes in all succeeding blocks (and therefore invalid PoWs). The immutability presents legal challenges, since data cannot be erased globally [2]; though one can simply delete potentially illegal data locally without diminished security guarantees [4].

[18] *Similar to the chained commit structure in the version control system Git.*

Each block, through its transactions, represents a change to the global ledger in that the coin balances of a subset of addresses are changed. Therefore, instead of storing a global ledger state of which address owns which amount of coins at time t with a blockchain of length N, this global state S_t is *implicitly* represented through the chain of blocks itself. To obtain S_t, one has to traverse the chain of blocks from the genesis block (=state S_0) and successively apply every transaction of each block B_i, f.a. $i = 0, \ldots, N$:

$$S_0 \xrightarrow{B_0} S_1 \xrightarrow{B_1} \ldots \xrightarrow{B_N} S_t \tag{1}$$

The term "state" already hints at the fact that Bitcoin (and related systems) can be equivalently understood as a massively replicated state transition system. We will go into the details on this interpretation in Section 2.3.2.1 when transitioning from Bitcoin to the more general system model of Ethereum.

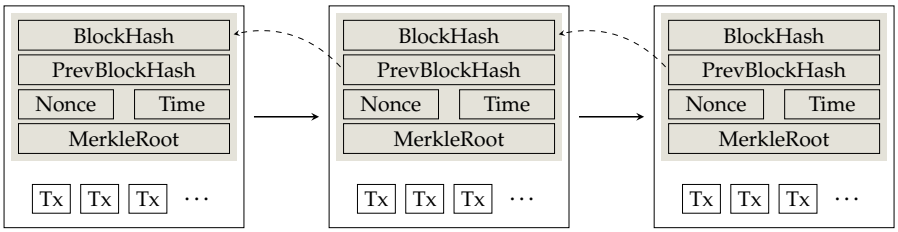


Figure 5: Blockchain and block structure (inspired by Fig. 1 in [257]).

2.3.1.3 *Analysis of the Mining Process*

Returning back to Alice and Bob, after broadcasting her transaction T in which Alice sends 5 coins to one of Bob's addresses, the transaction is eventually included by a miner in a block B. In fact, T is likely to be overheard by several miners, which include T into several pre-blocks $B_1, \ldots, B_n$; that is blocks without a valid PoW yet and in direct succession to tip of the blockchain, i.e., the last-known valid block, say, B_v. The miners are simultaneously competing to find suitable `nonce` and `time` fields such that the hash of their respective block B_i starts with the required number of leading zeros. As soon as a miner finds a PoW, it broadcasts the block B_i to all other peers in the network, which, after verifying the PoW, append the block to their local blockchains (i.e., B_v), updating the ledger. Since $B_1, \ldots B_n$ point to the same predecessor block, the other miners stop their search and start anew with B_i as the new tip of the blockchain, since only the *first* miner to find a successor block to B_v receives a monetary reward for finding the block.

Alice's transaction T is now, in block B_i, part of the blockchain and started to be considered part of the consensus, neglecting the eventual consistency through potential blockchain-reorganizations,

which are covered in Section 2.3.1.4. Assuming B_i (and therefore T) are considered sufficiently final (i.e., B_i is not going to be changed with high probability), this enables Bob to spend his 5 newly received coins by repeating the same process: (1) unlock the address by providing a valid signature, (2) broadcast the transaction T' to the network and (3) wait for T' to be included in some block B' and to eventually fulfill a heuristic for block finality.

How long do Alice and Bob have to wait until their respective transactions are mined into a block? In other words: how long does it take to find a suitable PoW for a block?

[19] *An approximation, of course. In practice, the time until a transaction is mined into a block also depends on the fees paid to the miners (higher fees → quicker inclusion into a block) and the number of transactions in the mempool.*

The PoW can be seen as a stochastic process[19], enabling insights in, e.g., the expected number of operations (and therefore the expected waiting time) to find a solution. Hash functions are approximations of random oracles [32] and strive to achieve a roughly uniform distribution of hash values. That is, each bit of the output hash $\vec{h} = (h_1, \ldots, h_n)$ has equal probability of $\frac{1}{2}$ of being 0 or 1; independently of the other bits. Assume the PoW is to find a hash $\vec{h}$ with d leading zeros, the probability to find such a hash by performing one hash operation is

$$\mathbb{P}[h_1 = 0 \wedge h_2 = 0 \ldots \wedge h_d = 0] \tag{2}$$

$$\overset{\text{independence}}{=} \mathbb{P}[h_1 = 0] \cdot \ldots \mathbb{P}[h_d = 0] \tag{3}$$

$$= \left(\frac{1}{2}\right)^d = 2^{-d}. \tag{4}$$

Let X be a random variable (r.v.) denoting the number of operations necessary to find a block. The process of finding a block can be seen as a sequence of Bernoulli trials with success probability 2^{-d} until the first success, therefore, X is geometrically distributed. In expectation, the number of operations to find a block is then[20]

[20] *The expected value of a geometrically distributed r.v. is $\frac{1}{p}$, where p denotes the success probability of the Bernoulli trial.*

$$\mathbb{E}[X] = \frac{1}{2^{-d}} = 2^d. \tag{5}$$

Hence, the number of operations required, and therefore the time to find a suitable hash, grows exponentially with the requirement of leading zeros, d. Equivalently, the time to find a solution to the PoW (and therefore a block) is inverse proportional to the invested computing power.

In Bitcoin, the requirement on the number of leading zeros (also called the *difficulty*) is periodically adjusted (roughly every two weeks [257]) to scale with the hashing power of the system to achieve a, relatively, constant rate of block creation. The target expected block interval in Bitcoin is 10 min, i.e., a new block is found every 10 min on average. At the time of writing, the difficulty in the Bitcoin network requires a block hash with 76 leading zeros, yielding an expected number of hash operations of roughly $7.56 \cdot 10^{22}$ — requiring an aggregated network hash rate in the ballpark of Exa hashes per second. On the one

hand, this massive amount of computational resources (and therefore energy) is considered a significant environmental problem [84, 150] which has sparked research efforts into alternative consensus mechanisms [241], e.g., Proof-of-Stake (PoS) [26] or hybrid trust models such as Ripple [16, 236] and Stellar [5, 162]. However, these alternative approaches require more trust assumptions than PoW-based consensus [67, 112], making it unlikely that Bitcoin will shift away from PoW. On the other hand, the invested computational resources can be seen as a high level of security instead of an environmental hazard. The more hashing power is invested in Bitcoin, the more resilient it becomes against so-called 51 %-*attacks*, which are covered in Section 2.3.1.4.

On a side note, the only motivation, when assuming economically rational agents, to participate in the mining process is the monetary reward received for successfully creating a new block. If the expected reward, i.e., the block reward discounted by the probability to find a block exceeds the cost to maintain and operate the necessary computing hardware, it is economically rational for an agent to participate in the mining process. As invested hashing power is "proportional" to robustness against 51 %-attacks, the price at which a cryptocurrency trades against fiat currencies (e.g., Euro, Dollar, etc.) is a driving factor of a cryptocurrency's resilience.

2.3.1.4 *Blockchain Forks*

Only blocks (and thus their transactions) with a valid PoW are accepted by other peers in the network, therefore, only valid blocks will lead to a change in the global ledger state. Validity criteria are, e.g., that blocks are below a certain size and contain no conflicting transactions, i.e., where the same input address is spent multiple times. For example, assume Alice wants to spend her 5 coins to Bob and Carol simultaneously by creating and broadcasting two transactions, as depicted in Figure 6. Each miner will only accept and include one into a new block, so that in the end either T *or* T' can be valid. However, in the description so far we assumed perfect data transmission between peers and no data races: all peers receive all information in a timely fashion and two blocks are not mined at the same time — assumptions that do not hold in practice. Peers may experience connection problems and/or two contesting blocks may be found almost simultaneously, leading to conflicts as to which of the contesting blocks should end up in the blockchain. Considering the example of Alice, Bob and Carol in Figure 6, it is possible that one part of the network knows only about T, whereas the other part only knows T' and both end up in respective blocks B_1 and B_2. The probabilistic nature of the PoW, as well as network delays etc. make it possible that one part of the network, P_1, learns about a block B_1 first, whereas the other part, P_2, learns about another block B_2 first — both

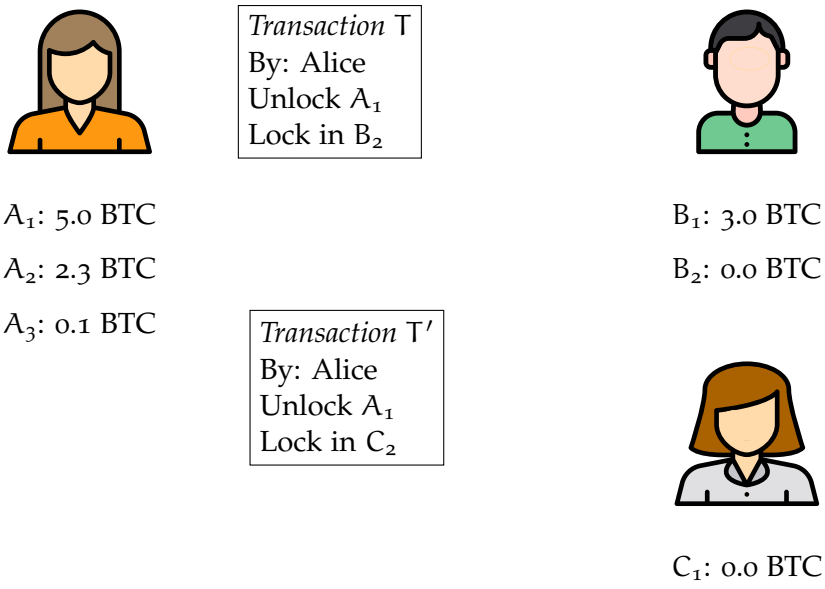


Figure 6: An attempted double spend of Alice (address A_1) to Bob and Carol. Icon attribution: [133, 134].

immediate successors of block B; see Figure 7. Since both blocks are valid each peer in P_i is "right" to believe that their respective block is the correct successor of B while the other block has to be discarded.

How are these conflicts resolved?

These so-called *forks* in the blockchain are resolved by the *longest-chain-rule*: new blocks are always attached to the currently longest chain of blocks[21]. In our scenario, proponents of B_1 (i.e., peers in P_1) would attach their subsequent blocks to B_1, whereas peers from P_2 would attach theirs to B_2. The rate at which new blocks are found is directly proportional to the invested computational resources (cf. Section 2.3.1). Let M_i denote the aggregate hash power of peers in P_i for $i \in \{1, 2\}$, respectively. As thoroughly laid out by Tschorsch and Scheuermann [257], unless both parts of the network have exactly equal computing power, one chain will grow faster, on average, than the other. Therefore, if $M_1 > M_2$, the chain appended to B_1 will eventually be longer than the chain appended to B_2 (and vice versa). In that case the longest-chain-rule ensures that *all* peers in the network will start to attach newly found blocks to the longest chain, thus resolving the fork by discarding the shorter chain. In the example depicted in Figure 7, B_2 is discarded, since the chain of B_1 is longer. Discarded blocks are considered invalid in that the state changes induced by their transactions are not applied; hence, the discarded transactions are treated as if they never occurred.

[21] *Note, that local, individual decisions by each peer eventually lead to a global convergence.*

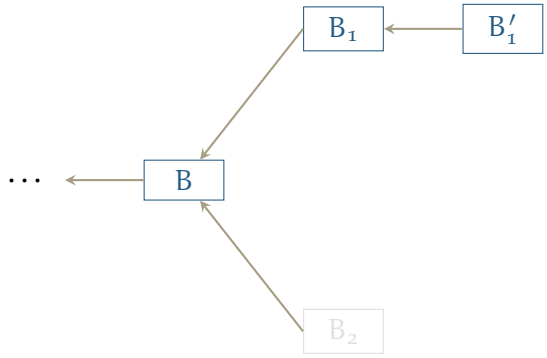


Figure 7: Example of a fork: B_1 and B_2 are found almost simultaneously. The fork is resolved since $B_1 \leftarrow B_1'$ is the longer chain.

2.3.1.5 *Transaction Finality & 51 % Attacks*

The combination of forks (involving a potentially unlimited number of blocks) and the invalidness of transactions not included in the main blockchain lead to the *eventual consistency* of Bitcoin's consensus [265]. Traditionally, Byzantine consensus protocols (such as PBFT [51]) strive for consistency, i.e., all nodes in the system agreeing on the same value within a bounded, finite time interval [265]. In Bitcoin, due to the presence of forks, only a weaker form of consistency, namely eventual consistency is reached: *eventually* forks are resolved, and all nodes agree on the same state again. When Alice performs a transaction to Bob she can, in theory, never be sure that her transactions remains valid and is not rendered invalid by a potential longer partial chain in the future. In practice, however, the probability of a fork declines with its length; most forks occurring during normal operation of the Bitcoin network are shorter than 4 blocks [81]. As a heuristic, people consider transactions final and valid after six blocks[22], i.e., there are six blocks on top of the block in which the transaction is situated.

Although the probability for accidental forks decreases exponentially in the length of the fork, an adversary might create a forked chain in secret. Upon release, honest node would switch to this chain as long as it is longer than the chain known so far — requiring the adversary to hold at least 51 % of the hashing power in the system. These kind of attacks are therefore referred to as 51 %-attacks. Such an attack enables an adversary Alice to double spend coins by first broadcasting a transaction T_B to Bob while retaining a secret chain with a transaction T_C spending the same funds to Carol. As soon as Bob accepts the transaction as valid and exchanges the goods (e.g., after six blocks), Alice releases her, longer, secret chain; rendering T_B invalid and depriving Bob of his deserved funds. Due to the immense hashing power invested into Bitcoin, obtaining 51 % of the networks hashing power is unlikely. However, 51 % attacks (and resulting double spends) have repeatedly happened with smaller currencies [18][23].

[22] *At Bitcoin's block creation rate roughly an hour.*

[23] *crypto51.app/ estimates and sorts cryptocurrencies by their estimated costs to carry out a 51 %-attack.*

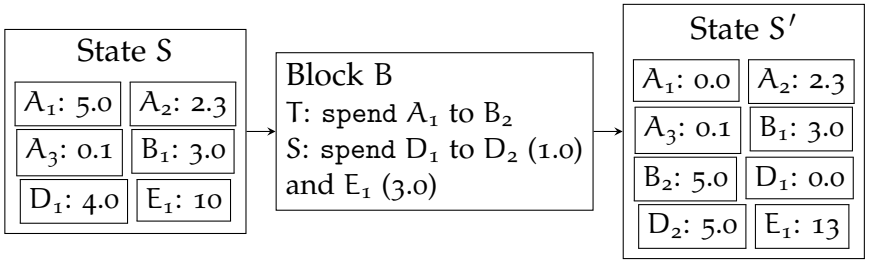


Figure 8: Bitcoin can be seen as a massively replicated state machine. This figure illustrates the state change induced by, among others, the transaction from Figure 4.

2.3.2 *Ethereum and Smart Contracts*

2.3.2.1 *From Transactions to Smart Contracts*

On the user-level, transactions are the transfer of ownership of coins from a set of addresses to another set of addresses. Looking from a global, ledger-level, a transaction (a block to be precise) advances the state of a massively replicated state transition system, as abstractly depicted in Figure 8. The current state of, say Bitcoin, consists of the individual coin balances of each address. This state, formed by the entirety of all coin balances, is replicated at every full node[24] and is advanced globally with every new block.

[24] *Only full nodes and miners store the whole blockchain, other peers utilizing* simple payment verification *may choose to trade security for performance by storing only parts of the chain and relying on full nodes for fetching those parts [202].*

In Bitcoin, a transaction is a small program, written in a stack-based, Turing-incomplete language which is executed by every full node in the network, as soon as it is included into a block. In theory, this allows for the creation of quite involved transactions, e.g., transactions only executed when k-out-of-n signatures are present; in practice, however, the expressiveness is purposely limited due to security concerns [257]. Most transactions follow these steps:

1. Sign the public key of the source address with the corresponding private key
2. Push the hash on the stack
3. Compute the hash of the public key in the blockchain
4. If equal: the source coins are successfully unlocked and locked with the destination script.

This restricted programming language is a deliberate design choice of Bitcoin — in theory there are no restrictions on the expressiveness of a transaction scripting language. The cryptocurrency Ethereum [271] chose the exact other direction as Bitcoin by using a full-fledged Turing-complete virtual machine instead of a simple transaction-script.

2.3.2.2 *Ethereum — A Payment System with Turing-Complete State Transitions*

Ethereum is the second largest decentralized P2P cryptocurrency, allowing users to exchange digital tokens on a blockchain, resembling Bitcoin. Unlike Bitcoin, Ethereum's transaction scripting language is Turing-complete and hence allows for arbitrary programs to be run when transactions are executed, allowing the creation of so-called *smart contracts*:[25] pieces of code that reside "on" the blockchain and whose execution can be triggered through transactions. The high popularity of Ethereum mainly stems from this possibility to create smart contracts and has lead to a massive ecosystem of developers and users. The most successful application of smart contracts is the ERC20[26] token standard [96], which is and was widely used to conduct so-called initial coin offerings — in theory a method of financing startups which has in practice also been used for extensive scamming [140, 279]. It allows the creation of fungible tokens whose exchange adheres to the same safety and security guarantees as the exchange of Ethereum inherent cryptocurrency, Ether.

[25] *Scholars have repeatedly pointed out that these programs are neither "smart" nor contracts and even Vitalik Buterin, one of Ethereum's founders, has expressed his regrets for choosing that terminology [46].*

Smart contracts are normal addresses which can also hold a balance of coins. If the code of a smart contract is to be executed, a transaction encoding the necessary parameters for execution is send to the contract's address. Similarly to normal programs, smart contracts can have internal variables, e.g., the total supply and each addresses' balance in the case of ERC20 tokens. These variables are stored in Merkle-Trie-like structures by every full node; the current root hash of these structures is part of Ethereum's block header and therefore part of the blockchain consensus.

[26] *ERC = Ethereum Request for Comment.*

Therefore, and in contrast to Bitcoin, the state of Ethereum does not only consist of coin balances at each address, but also the internal variables of each smart contract. Resembling Bitcoin, Ethereum can be seen as a massively replicated state machine, whose state is advanced *only* through the execution of transactions[27]. That is, transactions are gathered in blocks which also require a PoW[28] and yield a mining reward. Appending a block to the Ethereum blockchain (and therefore advancing the global state) involves executing all called smart contracts by each node in the system.

[27] *A common misconception among non-technicians, where smart contracts are often seen as permanently running programs, being able to constantly monitor and react to changes in the environment.*

Smart contracts are compiled into and stored in a dedicated assembly language, specifically designed for the use case of Ethereum, e.g., Keccak-256 hashes (the hash function of Ethereum) can be computed natively and methods from other smart contracts can be called, similar to libraries. Ethereum's assembly code is executed on the so-called *Ethereum Virtual Machine (EVM)*, though a transition to an Ethereum-flavored web assembly is planned [104]. While the Turing-completeness of Ethereum's programming language allows for maximum expressiveness, some restrictions are necessary to avoid unlimited computation times. To this end, each instruction on the EVM

[28] *Although Ethereum intends to transition from PoW towards PoS, cf. Section 2.3.1).*

has an associated cost called "gas". Hence, to execute a smart contract, the initiator of the respective transaction has to sufficient funds to cover the gas costs for the entirety of every executed instruction within the smart contract[29]. These gas costs and the maximum gas limit in a block bound the maximum number of instructions that can be executed. Restricting the number of possible instructions through associated gas costs is a reasonable design choice, since a smart contract is executed on *every* full node in parallel. Non-termination, e.g., through (non-)intentional infinite loops, would stall the entire network if not properly prevented. Determining suitable gas costs for each instruction that approximate the necessary computational load of an instruction is a problem in itself. Yang et al. [275], for example, demonstrated that the gas costs are too low for the expected runtime, especially on less powerful devices. Due to the undecidability of the halting problem [258] there is only a-posteriori knowledge on whether a transaction executing a smart contract has sufficient funds to cover the constract's resulting gas costs. In case a smart contract does not have sufficient funds for the execution to be finished, it is terminated and the state is rolled back before the execution of this particular smart contract while still consuming all provided gas.

[29] *I.e., the costs are sent with the transaction initiating the execution.*

The design choice for a Turing-complete smart contract language has been criticized in the past [138, 210] due to (1) verification issues and (2) the nature of smart contracts. Regarding (1), less expressive languages would be easier to verify and proposals for alternative programming language designs exist [210], although a transition towards a non-Turing complete language is unlikely, especially due to compatibility issues. Critics of the Turing-complete language point out that (2) most smart contracts do not need the expressive power of such a language, since they are relative small programs, lacking control flow characteristics offered by Turing-completeness in the first place [138]. However well-founded these arguments may be, it is unlikely that Ethereum will adopt another, less expressive, instruction set, in particular when switching to web assembly in the long term [104].

Smart contracts and the corresponding development ecosystem are the main reason why Ethereum is so widely used as an infrastructural layer for blockchain applications. First, these contracts are written in a developer-friendly dialect which bears similarities to JavaScript—a programming language many developers already know. Second, the Ethereum blockchain is running sufficiently stable and provides novice-friendly APIs that enable a manageable learning curve while also hiding the majority of the complexity under Ethereum's hood. Third, smart contracts can be interleaved similar to libraries, allowing the creation of highly complex systems, e.g., initial coin offering (ICO) tokens such as ERC20, DeFi (in particular stablecoins, cf. Chapter 5, [126, 222]) and DAOs [74].

2.4 THE BITCOIN OVERLAY

In the last sections we implicitly touched various aspects of the Bitcoin overlay — in this section we explicitly address (1) the requirements Bitcoin's overlay has to fulfill and (2) the differences to traditional P2P networking in terms of narrative and overlay structure. Although this thesis focuses on Ethereum as main application example, Bitcoin provides valuable insights through its well-thought-out design; allowing us to generalize the requirements an overlay for permissionless blockchain systems has to fulfill and to contrast those requirements to Ethereum's design choices and our exploitation thereof (cf. Chapter 3, where we also provide an in-depth analysis of Ethereum's P2P overlay). Note that the discussion here is necessarily limited, for an extensive survey on network layer aspects of permissionless cryptocurrencies in general (not only Bitcoin), the reader is referred to [208].

As already elaborated on in previous sections (cf. Section 2.3.1), the main mode of operation in Bitcoin is broadcast[30]: transactions and blocks originating at one node have to be distributed to all other nodes in the network. In this, Bitcoin differs from traditional P2P systems such as DHTs which are optimized for multicast: data items are requested and obtained by only a subset of peers. Furthermore, and in contrast to distributed hash tables (DHTs), Bitcoin's main optimization variable is not performance of data transmission but rather the robustness of information delivery against attacks and failures.

[30] *Other modes include bootstrapping newly joined peers and gossiping about neighboring nodes. On a side note: while "gossiping about neighbors" may seem perfectly fine for computer scientists, it is a clear anthropomorphization of technology [267].*

Bitcoin (as well as Ethereum and other permissionless cryptocurrencies) rely on the synchronicity and consistency of each peer's shared ledger: updates reach every peer eventually, inconsistencies are therefore short-lived [131]. If the network is partitioned, the shared ledgers may become out-of-sync (potentially purposely through an adversary), effectively leading to several isolated and diverging replicated state machines — one in each partition. The divergence is resolved through the longest-chain-rule as soon as the partition ceases to exist and connectivity is restored: the partition with the largest share of hashing power will eventually dominate the others, uniting them again on one chain and discarding all state changes that are not part of the dominating chain. Discarded transactions are treated as if they never occurred, enabling the double spending of funds, if an adversary where able to partition the network [131, 8, 168]. Adversarial overlay partitions are mainly enabled through so-called *Sybil* attacks.

2.4.1 *The Sybil Problem*

"On the Internet, nobody knows you're a cat"[31], this cartoon caption illustrates a fundamental problem in P2P networking: interactions on the Internet do not happen between entities of the real world

[31] *My sincerest gratitude goes to Marie Sofie Jacob for painting the sketch.*

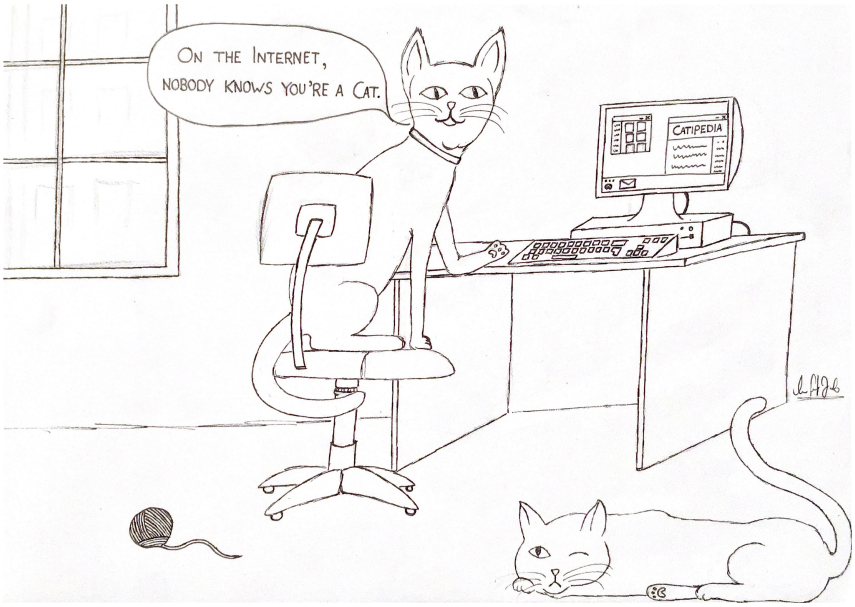


Figure 9: "On the Internet nobody knows you're a cat." vividly illustrates the Sybil problem. Based on a cartoon turned meme from Peter Steiner, originally published on July 5, 1993 in *The New Yorker*.

(e.g., cats), but rather informational abstractions of those entities, called *identities* [86, 139]. This enables real-world entities to liberate themselves from their inherent physical attributes (e.g., sex, looks or humanness in the case of the depicted cat), which may be one reason to communicate online [200]. In addition to physical attributes, the identity abstraction allows the possibility of freely choosing a digital persona and, therefore, social circles [157, p. 17f.].

While liberating, this free creation of identities also has problematic repercussions — most notably for our purposes is the ability of *one* entity to have *multiple* distinct identities. Examples include one person having multiple accounts in forums or social media sites. For other entities these distinct identities are nothing out of the ordinary and, in most cases, indistinguishable from honest participants for which "number of entities = number of identities" holds.

Leading from social to computer science, the ability of one entity to possess several identities is a core problem in P2P systems: an (adversarial) entity may start multiple instances of a P2P network's client software. It has to be noted that this process is less resource-intense in terms of human labor than creating and maintaining multiple accounts on social media sites and, additionally, automatable, i.e., posing a significant threat to P2P systems.

In a seminal work by Douceur [86], this adversarial behavior in the context of P2P systems has been defined as a *Sybil* attack [32] , replacing the previously occasionally used term "pseudospoofing".

[32] *The name Sybil attack refers to the book "Sybil" by Flora Schreiber [235] on the treatment of Sybil Dorsett for multiple personality disorder [86].*

Sybil attacks are especially problematic in voting/consensus and escrow systems [54], since one adversarial entity may seize voting power by spawning several identities. But also for P2P systems in general, whose mode of operation relies on the assumption that when selecting a subset of peers not all of them will be malicious, the Sybil attack poses a significant threat. In the case of DHTs for example, it enables attackers to perform denial of service attacks by strategically positioning Sybil nodes in the overlay which do not forward incoming queries or actively spread malicious information to bring lookups to a halt.

In the case of permissionless blockchain systems such as Bitcoin and Ethereum, the largest threats through Sybil nodes lies in partitioning the replicated state machine to either force a Denial-of-Service or double spend funds (cf. Section 2.4). These partitioning attacks are referred to as *eclipse attacks*, where an attacker isolates individual peers or even parts of the network such that all neighboring identities of a victim peer are operated by the adversarial entity[33], enabling the attacker to filter the information the victim sends and receives.

[33] *Technically, eclipse attacks can also be conducted by cooperating entities without Sybil identities. However, the large amount of necessary identities to successfully mount eclipse attacks in today's P2P networks makes this cooperation virtually impossible. Therefore, oftentimes an eclipse Attack requires Sybil identities.*

In many P2P systems, creating an identity is as easy as computing an asymmetric key pair, so what are potential remedies against such attacks? As shown by Douceur [86], it is impossible to fully thwart the possibility of Sybil attacks without (1) unrealistic assumptions on uniform resource distribution and simultaneous coordination of entities or (2) a trusted authority to vouch for the equivalence of identity and entity. Since (1) is not given in practice, a trusted authority is the only approach to fully solve the Sybil problem. However, oftentimes a trusted authority is not desirable, in particular when considering cryptocurrencies like Bitcoin which aim to be as decentralized as possible. Therefore, P2P systems in practice tie the creation of identities to some finite resource and/or implicitly and partially rely on the existence of some authority. Bitcoin has to defend against Sybil attacks on the consensus as well as the network layer and relies on both the scarcity of physical resources as well as the existence of a central authority in the process.

On the consensus layer, the process of finding new blocks can be understood as probabilistic voting mechanism. Instead of weighting each identities' vote equally, the computationally expensive task that has to be completed (PoW), effectively weighs each entities vote by its computing resources (cf. Section 2.3.1). Hence, the creation of additional identities to skew the voting process is futile, since only the physical resource determines the weight of ones vote and not the number of identities that are voting.

On the network layer, Bitcoin uses physical features and relies on the existence of a trusted authority to make Sybil attacks expensive. We take a deeper dive into Bitcoin's networking logic as it illustrates viable

real-world solutions against the Sybil problem while highlighting the limitations of said solutions at the same time.

2.4.2 *Bitcoin Overlay Structure and Sybil Protection Mechanisms*

In Bitcoin the so-called "AddrMan" (short for AddressManager) manages known IP addresses of other peers in a table-like data structure [75, 131].[34] In a nutshell AddrMan's design goal is simple: ensure that an adversary cannot overwrite the whole table with Sybil nodes. To that end, addresses are sorted into groups based on a combination of cryptographic hashing and a secret key[35]:

$$\text{targetbucket} = \text{Hash}(\text{IP} + \text{secret key}). \tag{6}$$

Thus, making the sorting (1) impossible to predict for an adversary due to the secret and (2) uniformly distributed, due to the hashing. Furthermore, AddrMan employs a two-stage storage, where the first stage only contains addresses the local AddrMan has heard about whereas the second stage holds addresses that AddrMan has successfully contacted before. Since buckets have a limited capacity, AddrMan randomly evicts an old entry — previously with a bias towards least recently seen addresses, which has been updated to random eviction with the work of Heilman et al. [131].

[34] *We refer to Bitcoin Core v0.19.1 in the following, but the general logic has been relatively stable and probably will remain similar in the future.*

[35] *An idealized representation of the actual implementation.*

2.4.2.1 *How does this "solve" the Sybil problem?*

The (uniform) randomness ensures that an adversary needs a significant number of Sybil nodes to have a reasonable probability to receive a connection from the victim node [131]. Additionally, since the mapping to buckets is based on the IP address, these Sybil nodes must come from a variety of different networks (/16 for IPv4, /32 for IPv6 [75, 76]) — otherwise Sybil nodes would be assigned to the same bucket, effectively competing for the same connection "slot". To successfully thwart Sybil attacks with this approach, the acquirement of many IP addresses in distinct subnets needs to be prohibitively expensive — which it is in practice. IP subnet allocation is centrally governed by the Internet Corporation for Assigned Names and Numbers (ICANN). In its function as a trusted authority it (1) maps IP subnets to entities and (2) establishes a notion of scarcity of the inherently digital resource of IP addresses. By mapping IPs from the same /16 subnet (for IPv4) to the same bucket as a Sybil protection mechanism, Bitcoin relies on the scarcity of IP addresses and therefore the ICANN as a trusted authority. As noted by Heilman et al. [131], despite these restrictions, botnets are still viable attack vehicles, as their inherent distribution circumvents IP subnet restrictions. Potential remedies aim towards maintaining entries of benign nodes in the buckets by only evicting offline peers, instead of favoring least-

recently-seen ones. As long as there is at least one legitimate peer in the buckets, a Sybil attack may deteriorate network performance but will not lead to an eclipse of the victim [131].

In addition to IP subnet restrictions and bucket eviction mechanisms, Bitcoin binds real-world resources as a Sybil protection measure since the expenditure of these resources further increases the cost of an attack. This can be illustrated by how Bitcoin decides if it should evict an old connection in favor for a new one [77]. The main risk associated with evictions is replacing an honest peer with an adversarial one, as stated in the comments:

> *The strategy used here is to protect a small number of peers for each of several distinct characteristics which are difficult to forge. In order to partition a node the attacker must be simultaneously better at all of them than honest peers.*

To that end, Bitcoin uses several metrics to protect connections from eviction; in particular it protects four nodes, respectively, which:

- have minimum ping times[36],
- are from distinct /16 subnets (/32 for IPv6),
- most recently sent transactions,
- most recently sent blocks and
- which have the longest uptime.

[36] *Moving average of measurements.*

The minimum ping time limits the scalability of a potential attack on several distinct Bitcoin nodes in parallel, since an adversary can not have minimum ping times to multiple geographical distributed nodes at once. To be considered a good connection candidate, an adversary has to spend his bandwidth to forward transactions and blocks, and invest computing resources for a prolonged period of time. Not only does this expenditure of real-world resource render Sybil attacks on Bitcoin expensive, it also forces the adversary to perform useful tasks for the network by distributing blocks and transactions when launching an attack. The necessary resources to conduct such an attack may outweigh the potential gains of the adversary, as will be discussed in section 3.9.

2.4.3 *Why is the Sybil problem still a problem then?*

Given the Sybil-protection mechanisms of Bitcoin one could argue that tying identities to real-world resources seems like a viable solution to the Sybil problem, rendering additional research superfluous. While intriguing at a first glance, the main problem with these approaches is the resource inequality between entities in a network, a fact already highlighted by Douceur [86]. Normal users tend to have less

resources than an adversary (which, even it is not true, is a reasonable assumption for a threat model). Hence, any precautions regarding the creation of identities (e.g., PoW) or requirements as to uptime, bandwidth and other computing resources should not be prohibitive for normal users[37]. Therefore, the Sybil problem is still very much a problem and, as Douceur [86] highlighted, generally unsolvable — it is only possible to raise the bar for an attacker by raising the costs associated with a Sybil attack.

[37] *For example, in S/Kademlia [29] the ID creation is tied to a PoW which should not take too long for users on weak devices. However, at the same time the PoW is hardly a challenge for a sufficiently equipped adversary. See [221] for a simple cost calculation for differently equipped adversaries.*

Reality is often not as bleak, however. The (implicit) adversary model used by Douceur [86] is situated on two extremes of a spectrum: on one hand, Sybil identities are prohibited by a trusted third party, on the other hand there is no prior knowledge on other nodes whatsoever. However, practice may lie between two these extremes when a node operator does have knowledge about some identities and their respective entities. This allows the creation of a few trusted connections for which the operator can be optimistic that no Sybil (or related) attack will be carried out.

2.4.4 *Scalability & Overlay Topology*

In summary, Bitcoin purposely sacrifices performance in terms of message delivery for robustness. In optimizing the overlay for robustness and security against Sybil attacks, Bitcoin resembles the TOR network [256], which deliberately routes packets over multiple hops to ensure the anonymity of a packet's origin; also sacrificing performance (end-to-end delay) in the process. Similar to Gnutella (cf. Section 2.1) Bitcoin utilizes an unstructured overlay and flooding to spread information on new transactions and blocks to every peer. Additionally, during the flooding of new transactions, they are not forwarded immediately to neighboring peers — instead a random Poisson-distributed delay is imposed. This so-called *diffusion*[38] of transactions aims at obscuring network layer identity of a transaction's issuer as well as to make it harder to assess the topology of the network as a whole [72, 207]. Bitcoin's overlay topology as well as individual nodes can be inferred to some degree through exploiting under-connectedness of clients [38], timing analysis [178, 207] and purposely crafting invalid transactions [73, 124]. However, these approaches have limited expressiveness [207], are prohibitively expensive [73] and/or are easily thwarted [124]. Nevertheless, individual Bitcoin clients do not seem to enjoy network level anonymity [38, 99, 204], regardless of trickling or diffusion [99]. Proposals for privacy-preserving data dissemination strategies [98, 204, 261] are currently not implemented in Bitcoin.

[38] *The original trickling mechanics have been replaced by diffusion in bitcoin core 0.13.0 [78], but the current approach serves the same purpose.*

As robust as the Bitcoin overlay may be, the deliberate choice of sub-optimal performance in terms of message delivery delay as well as the requirement of keeping all peers (and therefore the replicated state machine) synchronized impose a severe issue: scalability in terms

of transaction throughput. New blocks have to reach a sufficiently high share of all peers in the network, even ones with small bandwidth. Croman et al. [64] suggest a minimum threshold of 90 %, i.e., 90 % of peers should receive new blocks in time. This limits (1) the maximum size of blocks, which is 4 MB[39], and (2) the inter-block rate, effectively bounding the maximum possible rate of transactions per second that can be processed by the Bitcoin network at 27 transactions per second [64]. When the number of new transactions exceeds the maximum throughput for a sustained period of time, i.e., the network is overloaded, transaction fees (which are paid to miners) begin to rise, as miners preferably include transactions with the highest fees in their blocks [257]. Note, that this is not simply limited to Bitcoin, Ethereum exhibits the same problem and both cryptocurrencies have shown signs of congestion in the past [181, 270].

[39] *The original limit was* 1 MB, *which sparked significant controversies, two hard forks of Bitcoin [79, 80] and has been replaced by a block* weight *limit during the Segregated Witness update [163].*

The most promising remedy to the scalability problems of major cryptocurrencies are so-called *layer*-2-solutions, e.g., the Lightning [216] network for Bitcoin and the Raiden [206] network for Ethereum, which are out of scope of this thesis and are simply mentioned for the sake of completeness. Intuitively, parties establish a payment (or, more general: state-)channel to process payments off-chain and only initiate transactions in the Bitcoin network in the case of disputes or to open/close said state-/payment channels [125].

In the following we will dive deeper into eclipse attacks on P2P systems and give an overview of the literature on said topic, before moving on to our attack on Ethereum in Chapter 3.

2.5 ECLIPSE ATTACKS

Eclipse attacks on P2P systems are well-researched, with a vast body of literature concerning attacks and potential countermeasures [50, 56, 115, 239, 240]. There are several sub-forms of eclipse attacks, in its "purest" form, adversarial nodes block and filter the victim's view of the network, segregating it from other honest peers — analogous to solar eclipses (hence the name). To this end, an attacker occupies all in- and outgoing connection slots of a target node, as depicted in Figure 10: the overlay is considered as a graph, with nodes as vertices and connections among them as edges. The victim node t is exclusively connected to adversarial nodes a_1, a_2, a_3 — hence, any request issued by t to other honest nodes v_1, v_2 (or vice versa) is filtered, i.e., manipulated or simply dropped, by the attacker.

Other forms of eclipse attacks include localized and topology aware eclipse attacks [115] which exploit the overlay structure of DHTs such as Kademlia (cf. Section 2.2). Instead of individual peers, parts of the logical node ID space are poisoned with adversarial nodes. In filesharing networks such as KAD for example, this can be used to make certain search keywords (which also correspond to keys in

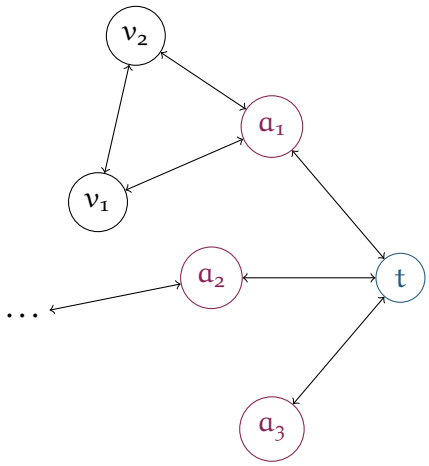


Figure 10: Illustration of an eclipse attack

the overlay) unreachable [147, 161], allowing an adversary to return arbitrary data to nodes searching for specific keywords.

An eclipse attack does not necessarily require a Sybil attack, a complot of several distinct adversarial nodes may also be conceivable, but is often preceded by one. Especially on modern P2P systems in the context of blockchain applications, the amount of distinct entities required for an eclipse attack necessitates a Sybil attack [131, 8, 168, 220]. As the Sybil problem is potentially unsolvable under realistic network conditions and in purely permissionless settings [86, 161], countermeasures against eclipse attacks can only raise the bar for an adversary instead of assuring a complete absence of attacks in the future. However, as, e.g., in the case of Bitcoin, the introduced countermeasures require a powerful adversary with control over a small botnet or ISP-grade resources [131].

Many countermeasures can be classified as (1) introducing some kind of trusted third party or (2) introducing randomness & redundancy in the peer selection and lookup process. For a detailed analysis of countermeasures, the reader is referred to [50, 57, 239]. Trusted third parties have been proposed to limit the creation of node IDs, e.g., by cryptographically signing them, rendering Sybil attacks virtually impossible, as these centralized parties ensure identity = entity [50, 238, 239, 245].

While effective at reducing the Sybil problem, centralized, trusted third parties are not desirable or practical in certain circumstances, especially in permissionless settings. In these situations, countermeasures can only increase the necessary resources an adversary has to invest for a successful attack. Popular are IP restrictions, e.g., by tying node IDs to IP addresses [168, 245], and/or by restricting the number of node IDs from one /8, /16 subnet in internal routing structures [75, 8, 147]. As discussed in Section 2.4.2, the reasoning is that IP addresses are scarce and obtaining many of them in distinct subnets

is challenging. To cope with powerful adversaries, e.g., in the case of (rented) botnets which consist of many diverse devices spread over a variety of IP subnets, performance is often sacrificed for robustness [29, 115, 213]. This trade-off entails replicating content lookups over multiple, disjoint paths [29, 213], or even letting lookups diverge, resembling a hybrid of structured search and flooding.

Transitioning from the general case of eclipse attacks on arbitrary P2P networks, what are the implications of eclipse attacks on blockchain systems such as Bitcoin and Ethereum? The main implication of these attacks are the ability of an adversary to double spend funds [131]. Normally, double spends are resolved by only including one of the two conflicting transactions into the global consensus. Even if blockchain forks occur, they are eventually resolved (cf. Section 2.3.1.4). If a victim is eclipsed, the conflicting forks can be maintained for a prolonged period of time, as the adversary filters and blocks the message exchange between victim and the rest of the network. Combined with some mining power on part of the attacker, the victim can be fooled into believing that the received transaction is valid, e.g., through mining a sufficient number of blocks on top of the doubly spent transaction, leading to an exchange of goods for the allegedly received money. As soon as the adversary receives her product, the eclipse attack is terminated and the deliberate adversarial fork of the blockchain at the victim is resolved — yielding funds and product for the adversary.

Beyond double spending, eclipse attacks have also been proposed to obtain a mining advantage, i.e., receive a larger share of the rewards than the computing power would entail [205]. However, the conclusions in [205] on combining stubborn mining with an eclipse attacks are debatable, as miners seem to be well-connected among themselves and to the Bitcoin network[40], rendering such an attack futile. In Ethereum, double spending eclipse attacks are not only possible on the native Ether-token, but also on other tokens, e.g., ERC20 and related standards. Furthermore, eclipse attacks can simply be used to achieve a DoS of smart contract execution, affecting the availability of applications.

[40] *https://bitcoinfibre.org/.*

2.6 BLOCKCHAIN APPLICATION STACK MODEL

As already hinted upon in the introduction, we study the robustness of blockchain applications in this thesis by dissecting them into different building blocks or layers which are then investigated individually. Specifically, from each building block we pick one popular representative system. This method has two advantages: (1) these building blocks are popular among several distinct blockchain applications, allowing us to transition beyond individual applications, and (2) in-

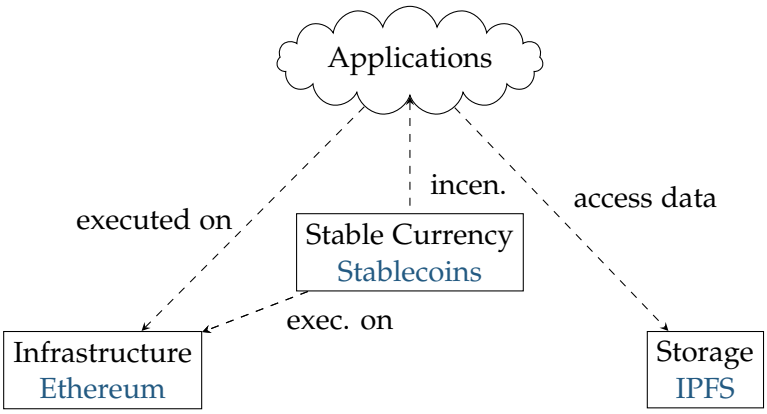


Figure 11: Blockchain application stack as well as the respective systems that we will investigate during the course of this thesis.

sights on building blocks are interesting in other contexts and not only limited to blockchain — in particular for IPFS on the storage layer.

We identified three essential blocks (or layers) that will be studied in this thesis, depicted in Figure 11 (the same as in the introduction): the infrastructural layer, distributed storage layer and currency stabilization. Interdependencies may occur between these blocks that may adversely affect the overall application, e.g., if services are incentivized through tokens and these token drop in value [145, 146]. An example would be the hash rate (and therefore the security against 51 % attacks) of a cryptocurrency which is directly dependent on the price of the respective coin. While an interesting topic and touched upon lightly throughout this thesis, the focus lies less on interdependencies and more on gaining an in-depth understanding of the individual blocks.

The infrastructural layer is what enables blockchain applications in the first place, so it is necessary to study a popular example. Ethereum is the subject of choice, since it is the most popular smart contract platform and therefore is the basis of many applications. Examples include ERC_i tokens ($i \in \{20, 223, 721, 777\}$) and ICOs thereof [223, 279], Aragon DAO [74] and, most importantly, Cryptokitties [270]. Applications building upon Ethereum (or other blockchains in general) rely on the guarantees the underlying system is providing, hence, any adversarial action on the infrastructure also impacts the respective applications. In the next chapter (cf. Chapter 3) we will see how eclipse attacks on Ethereum can impact applications, e.g., through double spending tokens and DoS.

Although platforms like Ethereum permit the storage of data in smart contracts, the costs are prohibitively expensive for even medium-sized files and folders, due to the redundant storage at every node in the network. Therefore, applications requiring to store more than just, e.g., internal variables, resort to external, distributed storage solutions. Thus, we argue that the storage layer, with IPFS as the subject of study as it is the most popular and mature system, is an important building

block for blockchain applications and beyond. To intertwine an application on, say, Ethereum with a storage system as IPFS, for example, one stores the hash and address of the corresponding data item on the blockchain[41] and the data itself on the storage network. Retrieving the data for the application then involves a lookup operation for the respective address and ensuring data integrity. The main concern arising through such a setup is the "health" of the storage network in terms of robustness, centralization, availability, et cetera. For example, a DoS attack on the storage network to bring down availability impacts the blockchain application as well. Similarly, if the data is essential for the application but hosted on a centralized server (or a distributed system that has become relatively centralized over time), the benefits of running the application on a blockchain are questionable. In Chapter 4 we take a deep dive into the networking- and content-side of IPFS, allowing us to reason about the health, robustness and degree of centralization of the network.

[41] *In the case of IPFS the address corresponds to the content, i.e., it is self-certifying (cf. Chapter 4).*

Lastly, incentives play an important role in decentralized, permissionless systems, as one cannot rely on the altruism of network participants alone to keep the systems running. Keeping Ethereum running and secured costs a significant amount of (computational) resources; data on IPFS has to be hosted and replicated. To this end, participants are economically incentivized to expend computational resources in Ethereum and related schemes are in planning for IPFS through a payment token for data hosting named Filecoin [154]. However, prices of cryptocurrencies are notoriously volatile. Therefore, we argue that value stabilization in cryptocurrencies in the form of stablecoins is another important building block for blockchain applications. Stablecoins strive to achieve the best of two worlds by combining (1) the stability of fiat currencies (e.g., USD, EUR), with (2) a permissionless setting like Bitcoin or Ethereum. Despite their use in payments, stablecoins are also essential for decentralized finance (DeFi) — complex financial derivatives on top of Ethereum [268]. In Chapter 5, we will see if stablecoins are able to fulfill their ambitious goals. To this end, we adopt an economic perspective to reason about stability, also in the presence of adversarial market forces, complemented with technical considerations.

3

ECLIPSE ATTACK ON ETHEREUM

3.1 OVERVIEW

A dependable and secure network layer is vital for blockchain systems, as they build on the assumption of equal information at every peer [131, 257]. As we have seen in Section 2.4.2, this assumption is violated if eclipse attacks are possible. In an eclipse attack (cf. Section 2.5), an adversary monopolizes the connections of a victim, effectively filtering the victim's view of the blockchain. This opens up attack vectors for, e.g., Denial-of-Service (DoS), double spending and stubborn mining [205]. In Section 2.4.2 we have also seen how Bitcoin quite successfully copes with the threat of Sybil/eclipse attacks by making such attacks costly to mount. Briefly summarized, Bitcoin establishes an unstructured overlay by randomly choosing peers from a large set of known nodes, such that no localized attacker can monopolize the set of known nodes [75] — demonstrating reasonable design approaches for building a robust overlay in the context of blockchain systems. Bitcoin is not immune to network layer attacks, of course, but it's overlay design makes them costly.

Ethereum, on the other hand, does not take into account the existing knowledge from Bitcoin's network layer and instead opts for a *structured*, Kademlia-based (cf. Section 2.2) construction. Since Kademlia was designed for efficient content lookup and data distribution instead of overlay robustness, vulnerabilities arise [8, 168]. These attacks, the first[42] by Marcus et al. [168] and the second by us [8], which is presented in this chapter, exploit the structure imposed by Kademlia to perform eclipse attacks on Ethereum peers with very little invested resources. In particular, our attack succeeds, in most cases, in a matter of hours with a newly-started client and requires only two IP addresses from distinct /24 networks.

Both attacks focus on *Go Ethereum (Geth)* [105], although other clients, e.g., Parity/OpenEthereum [153] seem to be vulnerable as well, briefly judging by their code. However, neither Marcus et al. [168] nor we [8] did verify this claim empirically, as a focus on Geth is

[42] *Notions as "first"/"second" imply a time-wise happens-before order of papers that was not given in this particular circumstance: Marcus et al. published their findings while we were in the midst of a publication.*

This chapter is based on previous collaborative work [8]. Understanding, finding and analyzing the potential attack vectors as well as planning the evaluation was done by myself (refined in extensive, insightful and incredibly helpful discussions with Martin Florian). Daniel Teunis mostly implemented and conducted the evaluation of the attack (Figures 17 and 18). As it's common, all authors contributed to the text of the paper; with some extensions and refinements by myself for this chapter.

reasonable: (1) it is the official reference implementation of Ethereum and (2) it is estimated to be used in roughly 76 % of clients [143].

Although both attacks target the structure imposed by Kademlia, they differ in execution, in that Marcus et al. [168] take a straightforward approach, while our idea is a bit more involved. Marcus et al. [168] flood an Ethereum victim node's discovery table with Sybil nodes [86] (cf. Section 2.4.1) directly after the victim has been restarted. This entails that any selection of peers for connection establishment only involves Sybil nodes. The discovery table is a Kademlia-style bucket structure, such that flooding this table simply involves filling each k-bucket with Sybils. The attack is lightweight, since generating a new node ID, and henceforth a new Sybil node, involves only an ECDSA key pair generation. As an answer to the discovered attack vector, Geth $\geqslant$ v1.8.0 introduces several countermeasures to increase the difficulty and necessary resources to flood the complete discovery table. One of these restrictions is the limitation of identities from the same /24 IPv4 subnet, which makes the attack of Marcus et al. [168] unattractive due to its high costs.

However, these countermeasures are not enough, and, as we show in this chapter, eclipse attacks on Geth $\leqslant$ v1.9.0[43] are still possible with very limited effort. Instead of overwriting the complete discovery table with Sybil nodes, for our attack, we subtly insert adversarial nodes with carefully selected node IDs, exploiting the interplay between Kademlia-based peer discovery and connection management. In particular, we exploit how Geth selects peers from the discovery table for establishing connections which are subsequently used for transmitting transactions and blocks. Despite the subnet restrictions implemented in Geth v1.8.0, we only need two IP addresses from distinct /24 subnets for a successful attack. Additionally, and in contrast to [168], we do not necessarily require a restart of the victim node when peer churn is high and existing connections will eventually be dropped, although a restart significantly accelerates the attack.

[43] *We responsibly disclosed our findings to the Ethereum foundation which led to (1) a bug bounty of 8000 points, yielding a place within the top 20 of Ethereum's bug bounty leaderboard (bounty.ethereum.org/) and (2) a personal exchange with the corresponding developers and subsequent hot fixes in Geth v1.9.0.*

Geth chooses new peers either by (1) directly selecting nodes from its discovery table or by (2) starting a Kademlia-style lookup to a random target, which yields new node contact information. We compromise both mechanisms in slightly different ways, but always exploiting the fact that node IDs in Kademlia are public, static and can be used to infer a node's bucket structure, i.e., which node ID will be mapped to which bucket. This allows us to generate suitable node IDs (= ECDSA key pairs) such that we are able to insert a limited number of Sybil nodes into the victim's discovery table, one Sybil node per bucket, with an activity pattern that favors these Sybils when new connections are set up through (1). For new contacts resulting from (2), lookup operations, we pre-compute a large number of node IDs and present tailored choices when queried during a lookup, effectively offering "better" (albeit false) peers than all honest nodes visible to the victim.

In the remainder of this chapter we dive into the details of this attack. Our contributions can be summarized as follows:

- The discovery, description and evaluation[44] of an eclipse attack on Geth versions $\leqslant$ v1.9.0 that exploits fundamental properties of Geth's peer discovery logic.
- A description and theoretical analysis of Ethereum's network layer management algorithms, based on an analysis of the Geth codebase; previously available information is scarce.
- A juxtaposition of possible and implemented countermeasures; both easy fixes to prevent the presented attack and ideas for tackling the fundamental challenge of securing Ethereum's overlay network.

[44] *On our own Ethereum node, without harming any other network participants.*

On a side note, we chose to name our attack the "False Friends Attack", due to the history of explaining these technical concepts to people without technical backgrounds. When working in an interdisciplinary environment, as the authors during the creation process of this attack, the language barrier between technicians and non-technicians impedes conversations on purely technical concepts. Hence, by presenting technicalities in terms of the real world, at least some knowledge can be transfered through this barrier. The most successful explanation appeared to be associating nodes on the network with people and referring to overlay connections as friendships. An Ethereum node is, therefore, looking for people to befriend (= establish overlay connections) while keeping a list of potential friends (= the discovery table) to call, when an actual friend leaves (= disconnects). The attack now involves creating fake friends in such a fashion that they specifically fulfill what the victim node is looking for in a friend (= crafting suitable node IDs for each bucket and the lookup operation).

In the following, after establishing related work (cf. Section 3.2) and the necessary background to Ethereum's network stack (cf. Sections 3.3 and 3.4), we present the false friends attack as well as a theoretical analysis in Sections 3.5 and 3.6. The chapter finishes with a real-world evaluation of the attack in Section 3.7, as well as a discussion of countermeasures and deliberations on the cost/benefit ratio of eclipse attacks real-world systems.

3.2 RELATED WORK

The literature on security considerations in peer-to-peer networks in general and Kademlia-based networks in particular is vast; in the following we focus on research most closely related to our attack, as an introduction to eclipse attacks in general was given in Section 2.5.

3.2.1 *Attacks on Kademlia-based networks*

The security of Kademlia [170] and its inspired implementations have been studied extensively [147, 161, 245, 264]. Steiner et al. [245] explore the space of possible attacks and implications whereas subsequent works focus on optimizations of these attacks [161, 264] and circumventing implemented countermeasures [147]. Most approaches require the ability to arbitrarily choose node IDs. Therefore, many mitigations focus on restricting the generation of node IDs by some form of trusted third party, e.g., through cryptographic signatures [29, 50, 97, 238, 239, 245].

Similar to our false friend attack where we insert carefully selected node IDs into the victim's discovery table, [264] present a low-resource approach to poison routing entries in the KAD network. Given multiple attacking nodes, the ID space is partitioned and routing entries are hijacked by spoofing messages. In Ethereum, message spoofing and arbitrary node ID choice are impossible, making our attack conceptually different, though closely related to previous attacks on the KAD network. Most notably, [161] also conjecture that a purely trustless countermeasure cannot exist, due to the fundamental problem of Sybil identities [86].

3.2.2 *Eclipse Attacks in Blockchain Systems*

Heilman et al. [131] were the first to study eclipse attacks on peer-to-peer blockchain systems, in particular Bitcoin. Eclipsing Bitcoin peers requires an extensive amount of IP addresses, comparable to ISP-grade resources or small botnets and the implemented countermeasures further increase these costs. However, despite the introduced countermeasures, eclipse attacks are still possible when exploiting BGP [15, 255]. In these attacks, an adversary hijacks routing prefixes to insert itself as a man in the middle within the communication of Bitcoin nodes. Due to the lack of encryption in the Bitcoin overlay, packets can not only be delayed or dropped but even tampered with. Routing attacks on BGP are facilitated, as there are strong centralization tendencies in hosting providers, e.g., Amazon AWS and Hetzner are popular places for hosting full Bitcoin nodes. Gervais et al. [116] incorporate previous security considerations and provide a quantitative framework to reason about the fundamental tradeoff between security and performance in PoW blockchains.

Eclipse attacks are not only possible on Bitcoin itself but also on payment channel networks, such as the Lightning network [125, 225, 228]. Synchronization and timely response times are especially important in layer-two protocols, enabling time-dilation attacks to steal funds by delaying the transmission of blocks to a victim in order to commit a fraudulent channel state to the blockchain [225].

For Ethereum, [117] describe an attack on the block synchronization mechanism. When an Ethereum peer misses a block, it will start a synchronization with exactly one neighboring peer. An adversary can leverage this behavior to indefinitely stall the synchronization or inject an adversarial chain of blocks.

As noted throughout this chapter, several eclipse attacks on Ethereum are described in [168]. Our approach differs since we do not fill the complete table with adversarial nodes instead insert node IDs with specific properties.

The detection of eclipse attacks has been studied for traditional P2P networks, see for example [57, 136]. These approaches aim at detecting malicious response patterns through diverging lookups [115] or through network topology assessments [57]. In the context of blockchain systems the current state of the discussion is unclear. Signatures of packet arrivals [11] and block arrival times [273] have been proposed as suitable detection mechanisms, while others have argued that the large variance of, e.g., block arrivals makes such detections susceptible to a high degree of misclassification [225].

3.3 BACKGROUND: THE ETHEREUM NETWORK STACK

In the following, we introduce the overall architecture of Ethereum's Peer-to-Peer network as implemented in Go Ethereum v1.8.0. Unlike similar descriptions in related works [143, 168], the information presented here uses a naming of high-level components that is more strongly aligned with the official Ethereum terminology.

The overall network architecture of Ethereum is summarized in Figure 12. Ethereum's network layer consists of four major components, namely: *discv4* for node discovery; *RLPx* as a secure transport layer; *DEVp2p* for session management on top of RLPx and the actual *Ethereum protocol* (*eth*) which runs on top of DEVp2p. The Whisper protocol (for decentralized applications) and the *Swarm* protocol (for decentralized file storage) are other subprotocols on top of DEVp2p.

DEVp2p not only provides the foundation for the Ethereum protocol and other application protocols, it also manages connections to other peers, the entirety of which forms the overlay on which blocks and transactions are distributed. Geth by default has a total of 25 TCP connections to other peers speaking the Ethereum protocol. Of these 25 slots, 17 are reserved for inbound connections (initiated by other peers), whereas the remaining 8 are allocated for outbound connections. In this case, inbound means that a remote peer sent a `SYN`-packet to start a TCP connection with the local peer. No further restrictions apply to inbound connections; if an inbound slot is available Geth simply accepts any connecting peer that supports the Ethereum protocol and operates on the same network (main, testing, etc.). The 8 outbound

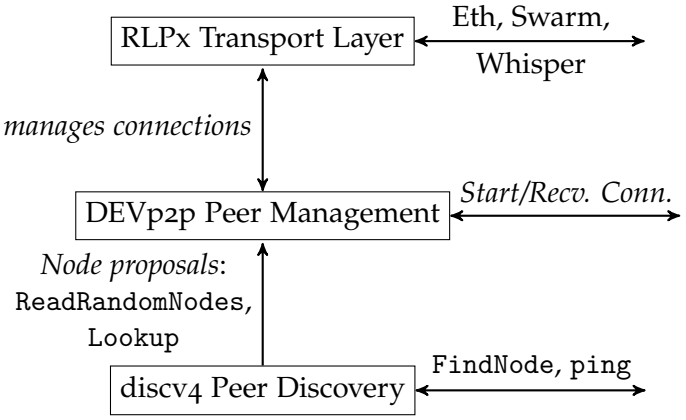


Figure 12: Overview of the Ethereum Network Stack.

slots are therefore especially important, as they are the most difficult ones obtain control of for an attacker mounting an eclipse attack.

In contrast to DEVp2p, the discv4 node discovery stores information about *all* node types in the overlay. This includes nodes without support for the Ethereum protocol (which is a perfectly valid case in the design logic of Ethereum's protocol stack). The discv4 node discovery is inspired by the Kademlia DHT (cf. Section 2.2) in that information about known overlay nodes is stored in a table separated into k-buckets.

This discovery table is used by DEVp2p to obtain outgoing connection candidates. Every time not all outbound slots are occupied, the DEVp2p peer management queries the discovery table in two distinct fashions depicted in Figure 13.

First, half of the *currently empty* slots (rounded down) are filled with a direct request to the discovery table via the function `ReadRandomNodes`. Second, the remaining slots are filled from the lookup-buffer, which holds the result of a Kademlia-like lookup to a random target ID. Note that this procedure is repeated *every time an outbound slot becomes available*. Therefore, Geth fills half of the *currently available* slots with each mechanism. Depending on the situation, this skews the distribution of outbound connections towards either mechanism. If only one slot becomes available at a time, the lookup-buffer is favored (`ReadRandomNodes` gets $\lfloor 0.5 \rfloor = 0$ slots). Otherwise, if two lookup-buffer slots become available repeatedly, `ReadRandomNodes` is favored in comparison to the lookup-buffer.

Our false friends attack exploits these two interfaces between node discovery and peer management. The discovery table therefore constitutes a particularly important component of Ethereum for our purposes, and deserves a closer look.

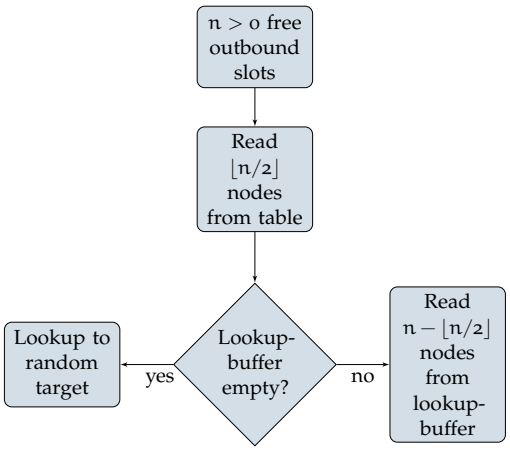


Figure 13: How outbound connections are established.

3.4 NODE DISCOVERY AND SELECTION

Although Ethereum's node discovery table largely resembles a Kademlia routing table, its sole purpose is to manage a set of known nodes which serves as a basis for establishing connections in DEVp2p. In [161] it is conjectured that Kademlia was chosen as a basis due to future plans to shard the blockchain, i.e., to partition it over the network for increased transaction throughput and scalability.

3.4.1 *Node IDs*

As in Kademlia, node IDs in Ethereum serve as public identifiers for each node in the Ethereum network. A node ID in Ethereum is a marshaled 512-bit ECDSA public key. However, distance computations only operate on Keccak256-hashes [36] of node IDs, effectively yielding node IDs with 256-bit length. When referring to node IDs in the following, this involves hashed ECDSA public keys. Node IDs are supposed to be static, as stated in the official Ethereum documentation:

> *Each node is expected to maintain a static private key which is saved and restored between sessions. It is recommended that the private key can only be reset manually [...].*[2]

It is easy to generate and use many different identities by creating ECDSA key pairs.

3.4.2 *Buckets and Log-Distance-Metric*

The buckets of Geth's discovery table hold up to $k = 16$ nodes each. Exactly as in Kademlia, the nodes in each bucket share a common

2 https://github.com/ethereum/devp2p/blob/6504d410bc4b8dda2b43941e1cb48c804b90cf22/rlpx.md

property: the distance, according to some metric, between their node ID and the local node's ID is the same. [143, 168] state that Ethereum uses the so-called *log-distance* metric. We argue that this metric is identical to the distance metric used to determine buckets in Kademlia. Marcus et al. [168] define the log-distance between two hashes be as $\lfloor \log_2(\overline{N}_1 \oplus \overline{N}_2) \rfloor$, or equivalently, 255 - the length of their common prefix — which is exactly how buckets are organized in Kademlia. Due to the uniqueness assumption of node IDs, this yields $|\{0, ..., 255\}| = 256$ possible distances.

In response to the eclipse attack by [168], Geth $\geqslant$ 1.8.0 restricts the number of buckets to 17, starting from the furthest distance of 255 to the minimum possible log-distance of 239.

The log-distance metric leads to a skewed distribution of nodes between buckets: most of the lower buckets are empty, since the probability to fall into a specific bucket decays exponentially with the associated distance [170].

3.4.3 *Entering a Bucket*

A local node learns of neighboring nodes either by receiving an unsolicited `ping` packet through a lookup operation. The lookup process also initiates a `ping`/`pong` exchange, which then triggers the node to be added to the discovery table. In any case, before a node enters the discovery table, a number of checks are performed which are depicted in Figure 14. Assume that the local node receives either a `ping` packet or a `pong` reply to a previously sent `ping`. Two cases are to be distinguished: first, the node may already be in its respective bucket; in this case it is simply moved to the first position. This induces a "least recently active" sorting of the nodes within a bucket [170], where activity simply means sending (responding to) a `ping`-packet. Second, in case the node is not already in a bucket, it is added if the bucket is not full. If the bucket is already full, candidate nodes are stored in a replacement list that stores up to ten nodes (as in the original Kademlia proposal [170]).

Every 5 s (on average), the last node of a random bucket is pinged and replaced with a random node from the respective replacement list if it fails to respond. In contrast to buckets, the replacement list is a simple FIFO queue that evicts the last entry every time a previously unknown node is added to the list. Last but not least, a node is only added to its respective bucket (or replacement list) if it meets certain IP address restrictions: Geth restricts the number of IP addresses coming from the same /24 subnet to two per bucket, and to ten in the whole discovery table.

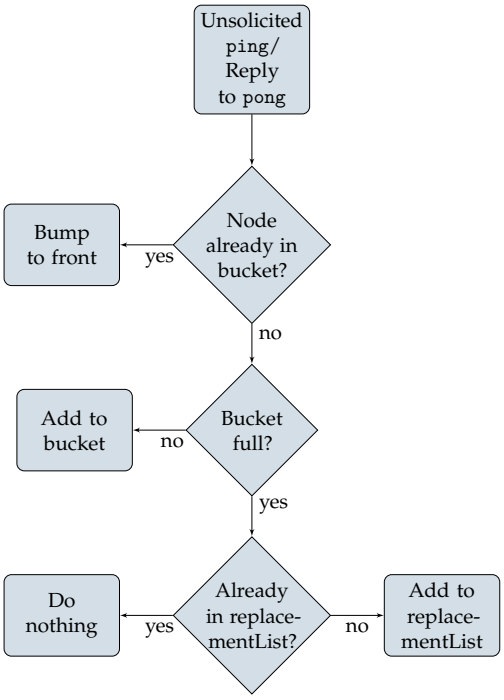


Figure 14: How nodes enter buckets.

3.4.4 *FindNode-Requests*

[45] *FindNode requests are also used to populate the discovery table and to resolve node IDs to IP addresses [168], which is outside the scope of this work.*

Lookups in Ethereum are used, among other things[45], to discover new peers. These lookups are performed iteratively by sending so-called *FindNode requests*, to which the recipient answers with a `neighbors` packet containing information about nodes from its discovery table.

The most important use of lookups in our scenario is to populate the so-called *lookup-buffer*. As already outlined, the lookup-buffer is one of two methods by which the DEVp2p subsystem finds new nodes to connect to. When the lookup-buffer is empty, Geth populates it by starting a lookup to a random target. That is, it sends a `FindNode` request to peers that are "close" to the random target. For lookups, Ethereum uses the plain xor metric instead of just the length of the common prefix. To illustrate, let $\overline{N}_1, \overline{N}_2$ be two node IDs and t a (random) target ID. For Ethereum (and likewise Kademlia), $\overline{N}_1$ is closer to t than $\overline{N}_2$ iff. $\overline{N}_1 \oplus t < \overline{N}_2 \oplus t$, where $\oplus$ denotes the bitwise xor operation and the result is taken as the binary representation of an unsigned integer. To ease notation, we define the abbreviation $<_t$ as $\overline{N}_1 <_t \overline{N}_2 :\Leftrightarrow (\overline{N}_1 \oplus t < \overline{N}_2 \oplus t)$.

The iterative lookup procedure to populate the lookup-buffer is visualized in Algorithm 1. First, a random target ID t is chosen. Subsequently, all known peers from the discovery table are sorted according to $<_t$, effectively yielding the 16 peers that are closest to the random target t. In a next step, a `FindNode` request is sent to each of these 16 peers, asking them for their respective neighbors that

are closest to t. If successful, each queried peer will answer with a `neighbors` packet containing up to 12 peers (1280 byte). All received neighbors are combined, sorted by $<_t$, and again restricted to the 16 peers closest to the random target t. This yields the result set of the first round. This process is iterated until the result set eventually stabilizes. If a result set contains the same peers as the result set of the previous iteration, the procedure terminates.

Algorithm 1 Populate Lookup-buffer

$t \leftarrow$ random node ID
$N_0 \leftarrow \{16 \text{ closest known peers to } t\}$
loop
  for $o_i \in N_0$ **do**
    $F_i \leftarrow \{\text{closest peers of } o_i \text{ to } t, \text{ as returned by } o_i\}$
  $N \leftarrow N_0 \cup \bigcup_{i=0}^{15} F_i$

  $N_1 \leftarrow \text{sort}(N, t)[0:15]$ // 16 closest to t
  if $N_0 = N_1$ **then return** N_0
  else
    $N_0 \leftarrow N_1$

3.5 THE FALSE FRIENDS ATTACK

After having established the necessary background in the previous sections, we now describe the details of our *false friends* attack, with an in-depth analysis of the attack following in Section 3.6.

To eclipse a victim, its 8 slots for outbound connections as well as the 17 slots for inbound ones have to be filled with adversarial nodes. The inbound connections slots can easily be filled since Geth does not impose any restrictions on inbound connections. Hence, it suffices to start multiple Geth instances on different ports and configure them to repeatedly connect to the victim. Due to the lack of restrictions, *one* moderately powerful host with one IP address is enough to fill the inbound slots. Note that these Geth instances do *not* need to actively participate in block and transaction distribution.

To fill the outbound connection slots, we have to make sure that only adversarial nodes are proposed to the DEVp2p peer management via the two mechanisms (cf. Section 3.3). Whereas [168] fills the whole discovery table with Sybil nodes to ensure that only adversarial nodes can be proposed to the peer management, we achieve the same result with only one Sybil node per neighbor table bucket. It suffices to have one Sybil node in each bucket of the neighbor table to make sure that only adversarial nodes are returned to the peer management. This effectively circumvents the implemented countermeasures and still needs very little resources (two IP addresses in distinct /24 subnets).

In contrast to [131, 168] we do not necessarily require a restart of the victim node for our attack to be successful, though it speeds up the attack. In both cases, an adversary has to wait until existing connections are terminated for other reasons, such as timeouts. Our measurements (discussed in Section 3.7) indicate that connections on the Ethereum network are rather short-lived and a successful attack against a non-restarting victim node is, in principle, possible.

Our attack is facilitated by the fact that countermeasures 2 and 3 from [168] were not implemented in Geth. Countermeasure 2, a fixed mapping between the IP address and ECDSA key would raise the requirements for a false friend attack to 25 unique IP addresses (one for each connection slot). While moderately increasing the necessary resources for an adversary, the impediment for the network is significant, since multiple Geth instances behind a NAT would be impossible. Countermeasure 3, making the mapping of IDs to buckets in the neighbor table secret, is a viable mechanism to drastically raise the bar for an attacker. In essence, this proposal is similar to our proposal of eliminating Kademlia altogether (cf. Section 3.8) However, as one would lose the benefits of Kademlia in case routing ever becomes relevant, the developers chose not to implement countermeasure 3 nor followed our proposal.

3.5.1 *Taking Over `ReadRandomNodes`*

The function `ReadRandomNodes` returns (per default) at most 4 nodes from the discovery table, which are then used by the peer management to establish outbound connections. Most importantly, `ReadRandomNodes` only returns the *head* of randomly chosen buckets. Presumably, this design choice is due to the implicit sorting by activity within a bucket (cf. Section 3.4): peers in the front of a bucket are more active and/or have a better latency than the others and are therefore favorable to connect to. This behavior can easily exploited by an adversary, since the sorting by activity merely requires the adversary to regularly send a `ping`-packet to stay ahead of the other peers. Therefore, it is sufficient for an attacker to populate each bucket with *one* node instead of the whole discovery table. To this end, the adversary repeatedly generates new ECDSA key pairs, computes the node ID and checks whether this particular ID is mapped to the desired bucket.

Current versions of Geth maintain 17 buckets and implement an IP-based restrictions such that at most 2 nodes from the same /24 subnet can be included in the same bucket and at most 10 nodes from the same /24 subnet can be in the whole discovery table. With these current properties of Geth, only *two* IPs from distinct /24 subnets are necessary for successfully compromising `ReadRandomNodes`.

3.5.2 *Exploiting the lookup-buffer*

The lookup-buffer is the second source used by DEVp2p to get potential peers to connect to. It is populated with a Kademlia-like iterative lookup of a random target ID. To this end, the local node sends `FindNode`-Requests with a random target to those nodes from the discovery table that are closest to that target (cf. Section 3.4.4). From the received node set, the 16 closest nodes are used to populate the lookup-buffer, sorted by their distance to the target. DEVp2p then partially fills the open outbound connections slots by going through the lookup-buffer from the top (i.e. minimum distance).

To fill the lookup-buffer with adversarial nodes two steps are necessary: First, an adversarial node must be queried during the lookup-process. Second, the node IDs returned by the adversary must be smaller than all other node IDs returned during the lookup. The first step is *always* given when there is an adversarial node in each bucket: the xor distance is mainly influenced by the length of the common prefix and each bucket stores node IDs with a specific common prefix length. Therefore, an adversarial node in each bucket ensures that the attacker is always queried during a lookup (cf. Section 3.6.2 for a detailed analysis).

The second step can easily be solved by generating sufficiently many node IDs. Since node IDs are hashed ECDSA keys, they are uniformly distributed over the ID space; hence, the more node IDs we generate, the higher the chances to be smaller than the rest of the returned IDs. In the end, by choosing the number of pre-computed keys high enough, it is very likely that all of our 16 closest IDs are closer to the target than any ID naturally occurring in the Ethereum network.

3.6 ANALYSIS OF THE FALSE FRIENDS ATTACK

In the following we analyze the mechanics of our false friends attack. We compute the expected number of necessary key pair generations and for entering every bucket and to exploit the lookup-buffer. Furthermore, we study the probability of receiving `FindNode`-Requests with different hypothetical bucket sizes.

3.6.1 *Entering a Bucket*

Our false friend attack requires one adversarial node in each bucket. The question arises how many key pairs we have to generate and how much time it takes to do so.

Since SHA256 is a cryptographic hash function, we assume node IDs to be uniformly distributed [100], i.e., each bit has a probability of $\frac{1}{2}$ of being 0 or 1. Therefore, each ECDSA key pair generation with subsequent hashing corresponds to fair coin tosses repeated

independently of each other. Let $\overline{N}$ be the hashed node ID of the victim node, i.e., $\overline{N}$ is fixed. Then, for some generated hashed node ID, say $\overline{h}$, the probability that the first bit of $\overline{h}$ is equal to the first bit of $\overline{N}$ is $\frac{1}{2}$. Recall that the log-distance metric measures the length of the common prefix between $\overline{N}$ and $\overline{h}$ (cf. Section 3.4.2). Hence, with probability $\frac{1}{2}$, the two hashes differ at the first bit, which corresponds to a log-distance of 255. For subsequent buckets the concept is similar: the probability to have a log-distance of 254 is $\frac{1}{4}$ since we have to be equal in the first *and* second bit, i.e., $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. In summary, the probability to have a specific log-distance to a given target hash is

$$p := \mathbb{P}[\text{log-distance}(h_1, h_2) = i] = 2^{i-256}. \tag{7}$$

Changing perspective, we can now calculate the expected number of necessary key pair generations to fall into a specific bucket. Finding a key pair for a desired bucket, or equivalently, a desired log-distance, can be modeled as a series of independent Bernoulli trials until the first success. Each key pair generation is a Bernoulli trial with success probability p (from Equation (7)). Repeatedly performing Bernoulli trials and stopping at the first success yields a geometric distribution. Therefore, generating key pairs for a specific bucket can be modeled as a geometric distribution, with expectation

$$\mathbb{E}[\#\text{ key pair generations for log-distance } i] = \frac{1}{p} = 2^{256-i}. \tag{8}$$

For example, to generate an ID with the (in Ethereum) lowest possible log-distance of 239 one would, on average, need $2^{17} = 131072$ key pair generations and hash operations. Generating a node ID for every bucket requires an average number of operations of

$$\sum_{i=239}^{255} 2^{256-i} = \sum_{i=1}^{17} = 2^i = 262142. \tag{9}$$

Note that these node ID generations need to be performed only once per victim node. Furthermore, in Section 3.7.1 we will see that even on a moderately powerful machine, 35000 node IDs can be generated per second.

3.6.2 *Computing the Probability to Receive a `FindNode`-Request*

In the following we analyze how probable it is to be asked during a `FindNode`-Request round. Recall from the previous section that we insert an adversarial node ID into each bucket for our false friends attack. Hence, searching for the 16 closest neighbors will *always* return at least one attacker-controlled node ID.

One could imagine that a simple countermeasure to the attack in Geth v1.8.0 is to simply increase the size of the buckets to hold more than $k = 16$ nodes. In the following we analyze the probability for an attacker to receive a `FindNode`-Request in different scenarios. As we will see, increasing the bucket size does not introduce significant hurdles for an adversary.

For the lookup-process, the victim generates a random target ID, say D, and computes its k closest neighbors to D. "Close" is defined in terms of the simple xor metric; for two IDs a, b the distance is defined as $d(a, b) := a \oplus b$, taken as integer. Under the xor metric, node IDs that have a longer common prefix D are thus considered closer than ones with a shorter common prefix. Each bucket partitions the binary tree of node IDs into branches by their common prefix. Therefore, the closest neighbors to D are the ones in D's bucket.

We can assume that node IDs are uniformly distributed in $\{0, 1, \ldots, 2^{256} - 1\}$ since they are hashed public keys with 256 bit length. Without loss of generality, let us map node IDs to be within $[0, 1]$. As a simplification, assume them to be continuously and uniformly distributed on said interval.

On a side note, in the following we refer to node IDs and distance between them as if they were real numbers on the unit interval with the <-relation — simply to ease the understanding. This does not impede analytical exactness: apart from the assumption of continuously distributed node IDs, xor operations and the normalization induce permutations of node IDs but to not alter the general structure. The xor distance metric is unidirectional [170], i.e., for a fixed x and fixed distance c, there exists exactly one y s.t. $d(x, y) = c$. Therefore, since node IDs are uniformly distributed, so are the distances to a specific target.

To illustrate, consider a lookup target ID D and node IDs $X_1, \ldots, X_m$, for a moment not normalized but as binary strings of length n. Each possible binary string is equally likely: $\mathbb{P}[X_1 = x] = 2^{-n}$. Hence, for a fixed D:

$$\mathbb{P}[d(D, X) = c] = \mathbb{P}[X = x] = 2^{-n}, \tag{10}$$

where x is the ID with $d(D, x) = c$.

To summarize this side note, we can omit the transformation induced by the xor distance and treat node IDs as numbers in $[0, 1]$ with the normal, well-known relations between them when talking about distance.

3.6.2.1 *Situation with Larger Buckets*

Let $D \in [0, 1]$ be the uniformly random target ID chosen by the lookup-process and $Y_1, \ldots, Y_N$ be the IDs of honest nodes stored in the bucket

of D, say b. We consider the case $k > N > 16$, i.e., the bucket is at its capacity limit with adversarial and benign nodes.

Assume for the moment that the adversary has exactly *one* node ID in b, with ID Z. Although the $Y_1, \ldots, Y_N$ share a common prefix, their suffix is distributed uniformly at random because node IDs are hashes. For an attacker to receive a `FindNode`-Request it suffices to be smaller (w.r.t. D) than the 17-th closest ID, i.e., the 17-th *order statistic*. We denote the ordering of IDs Y_i with respect to D as $Y_{(1)} <_D Y_{(2)} <_D \ldots <_D Y_{(N)}$, where $Y_{(1)}$ is the node with minimum distance to D. It now remains to compute the following probability:

$$\mathbb{P}[Z < Y_{(17)}]. \tag{11}$$

The density of Z is $f_Z(z) = 1$, due to its uniformity. Similarly, let $f_Y(y)$ denote the density of some Y. To compute Equation (11), we have to consider the joint distribution of Z and $Y_{(17)}$ in the cases where $Z < Y_{(17)}$. Note that Z and Y_i are independent. We then obtain for any independent Y:

$$\begin{aligned}
\mathbb{P}[Z < Y] &= \int_{y=0}^{1} \int_{z=0}^{y} f_{Y,Z}(y,z)\,dz\,dy && (12)\\
&= \int_{y=0}^{1} \underbrace{\int_{z=0}^{y} f_Z(z)\,dz}_{=y} f_Y(y)\,dy && (13)\\
&= \int_{y=0}^{1} y \cdot f_Y(y)\,dy && (14)\\
&= \mathbb{E}[Y]. && (15)
\end{aligned}$$

It is well-known that the order statistics of uniform variables are Beta-distributed [114], i.e., $Y_{(l)} \sim \mathrm{Beta}(l, N+1-l)$. Inserting that into Equation 15 we get

$$\mathbb{P}[Z < Y_{(l)}] \overset{(15)}{=} \mathbb{E}[Y_{(l)}] \overset{\text{Beta distr.}}{=} \frac{l}{N+1}. \tag{16}$$

In general, the attacker can have multiple, say $a \in \mathbb{N}$ nodes in the bucket b. To get queried, *at least one attacker ID* has to be within the closest nodes. Denote the adversarial IDs by $Z_1, \ldots, Z_a \sim U[0,1]$. Then we obtain:

$$\begin{aligned}
&\mathbb{P}[\text{At least one attacker ID within } l \text{ closest}] && (17)\\
&= 1 - \mathbb{P}[Z_1 > Y_{(l)} \wedge Z_2 > Y_{(l)} \wedge \ldots Z_a > Y_{(l)}] && (18)\\
&\overset{\text{i.i.d.}}{=} 1 - \mathbb{P}[Z_1 > Y_{(l)}] \cdot \ldots \cdot \mathbb{P}[Z_a > Y_{(l)}] && (19)\\
&\overset{(16)}{=} 1 - \left[1 - \left(\frac{l}{N+1}\right)\right]^a. && (20)
\end{aligned}$$

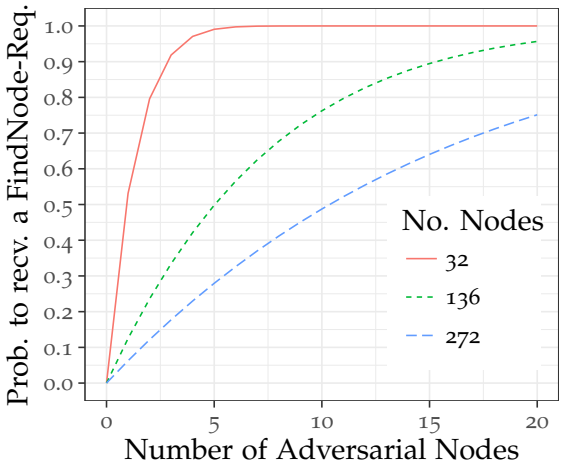


Figure 15: Probability for a single bucket that an adversary gets queried with a `FindNode`-Request for a given number of adversarial nodes in the bucket.

Intuitively, the more adversarial nodes there are in bucket b, the more unlikely it becomes *not* to get queried during the lookup-process.

Figure 15 shows the result of Equation (20). It depicts the probability for the adversary to receive a `FindNode`-Request over the number of adversarial nodes in bucket b, which is assumed to be at full capacity, i.e., $k = N + \alpha$[46]. For the number of nodes per bucket we consider three cases:

[46] *In Figure 15, the numerator of Equation (20) is to this end adapted to* $N + 1 - \alpha$.

1. $k = 32$, double the size of current buckets.
2. $k = 136$, i.e., the size of a bucket corresponds to half of the current size of the complete discovery table.
3. $k = 272$ which corresponds to the maximum size of the current discovery table (17 buckets à 16 nodes each).

3.6.3 *Filling the Lookup-buffer with Pre-Computed Node IDs*

Recall, that in order to take over the lookup-buffer, we identified two necessary steps: First, an adversarial node must be queried during the lookup-process. In a second step, the node IDs returned by the adversary must be smaller (with respect to the random target) than all other node IDs returned during the lookup. Intuitively, this can be ensured by pre-computing a large number of node IDs, since each ID generation corresponds to a draw from the uniform distribution. Every draw has the same probability to be smaller than any other node ID on the Ethereum network. Therefore, the more node IDs we generate, the more likely this event becomes. The question that remains is the following: how many ECDSA key pairs should an

adversary generate in advance to almost always return the smallest node ID?

To model this scenario, assume all other nodes already replied to the `FindNode`-Request with target D, which yields node IDs (sorted w.l.o.g) $0 \leqslant X_1, < X_2 \ldots < X_m \leqslant 1$. [47] For a new node ID to be the minimum, it has to be smaller than the minimum of the honest IDs, which is X_1. Therefore, it suffices to find the distribution function of X_1 and use Equation (15). The minimum of independent uniform random variables has the following distribution function:

[47] *Technically, the sorting is w.r.t. to* $<_D$. *As outlined before, the xor operation only induces a permutation of the* X_i, *preserving the uniform distribution. We therefore purposely omit the xor operation for the ease of understanding.*

$$\mathbb{P}[\min\{X_1, \ldots, X_m\} \leqslant x] = 1 - (\mathbb{P}[X_1 > x] \ldots \mathbb{P}[X_m > x]) \tag{21}$$

$$= 1 - (1-x)^m = F_{X_{\min}}(x), \tag{22}$$

with the density $f_{X_{\min}}(x) = m(1-x)^{m-1}$. Let $Y \sim U[0,1]$ represent a node ID generation and let $X := \min\{X_1, \ldots, X_m\}$ Using Equation (15) and integration by parts we obtain:

$$\mathbb{P}[Y < X] = \int_{t=0}^{1} t \cdot m(1-t)^{m-1} dt \tag{23}$$

$$= \underbrace{[-t \cdot (1-t)^m]_0^1}_{=0} + \int_{t=0}^{1} (1-t)^m dt \tag{24}$$

$$= \left[-\frac{1}{m+1}(1-t)^{m+1}\right]_0^1 \tag{25}$$

$$= \frac{1}{m+1}. \tag{26}$$

In other words, every generated node ID has a chance of $p := \frac{1}{m+1}$ of being the minimum node ID. Therefore, repeating the process of generating node IDs again yields a Bernoulli trial with success probability p.

In reality, we do not know the other node IDs before we start generating our own, meaning that whether a draw was successful cannot be determined. Instead, an adversary pre-computes a large number of IDs and simply returns the smallest ones with respect to the random target, if she receives a `FindNode`-Request. Still, we can bound the probability that *at least one* of our draws is smaller than the minimum returned by the honest nodes. Let there be m node IDs in the Ethereum network and n pre-computed node IDs by the adversary. For convenience, we define $Y_{\min} := \min\{Y_1, \ldots, Y_n\}$ and $X_{\min} := \min\{X_1, \ldots, X_m\}$.

$$\mathbb{P}[Y_{\min} < X_{\min}] = \mathbb{P}[Y_i < X_{\min} \text{ for at least one } i] \tag{27}$$

$$= 1 - (\mathbb{P}[Y_1 > X_{\min}] \ldots \mathbb{P}[Y_n > X_{\min}]) \tag{28}$$

$$= 1 - (1 - \frac{1}{m+1})^n \tag{29}$$

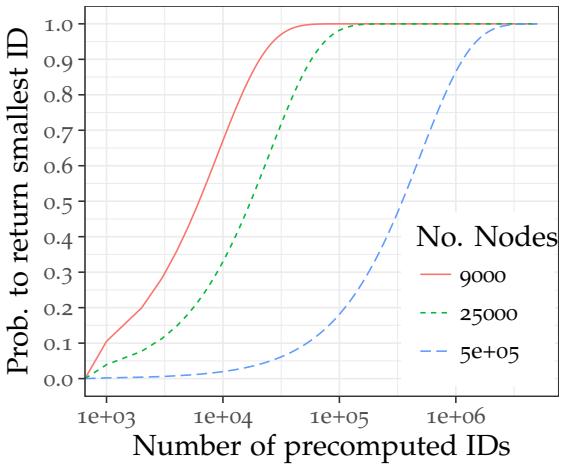


Figure 16: Probability that the lowest node ID is returned, depending on the number of honest nodes in the network.

The resulting probability is depicted in Figure 16 for different choices of n, the number of node IDs in the network. It can be seen that the more honest nodes there are, the smaller the probability for an adversary to have the minimal node ID becomes.

The red (solid) and green (short dashed) lines depict a situation where every node ID would correspond to an actual node on the Ethereum network. The red (solid) line shows the probability for $n = 9000$ nodes, as reported by Ethernode[48]. The green (short dashed) line corresponds to $n = 25000$ nodes in the network, the sum of all approaches discussed in [143]. The blue (long dashed) line corresponds to an upper bound on the number of node IDs of $n = 5 \cdot 10^5$ nodes. In all cases, however, $5 \cdot 10^6$ pre-computed ECDSA key-pairs are enough to return the minimal node ID almost certainly.

[48] https://www.ethernodes.org/network/1

Note that we have to distinguish between node IDs in the discovery table and actual nodes on the main Ethereum network. The Ethereum protocol is running concurrently with other protocols on the same communication channels and packet structures; therefore the number of node IDs is ten times larger than the number of Ethereum nodes at roughly $3 \cdot 10^6$ node IDs [143]. In case of returning the smallest ID for a `FindNode`-Request, this behavior slightly raises the bar for an attacker.

3.7 EVALUATION

We evaluated the previously described concepts using a victim node deployed specifically for this task. The victim had the latest Geth version from Github (v1.8.20) and was connected to the Ethereum main network.

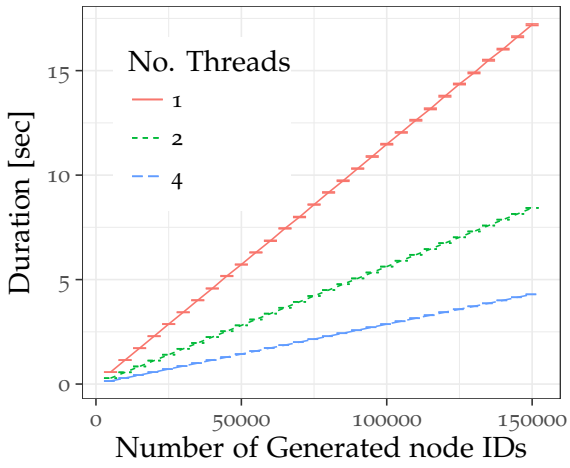


Figure 17: Mean duration of node ID generation with different numbers of parallel threads. The (very small) error bars show the 95% confidence interval for 100 runs.

3.7.1 *Pre-Computing Node IDs*

The generation of node IDs is essential for placing a node in each bucket and to ensure that the lowest ID is returned during the lookup-process. Therefore, we measured the calculation time for new ECDSA key pairs and the corresponding hashes. Our measurements were conducted on a system with an Intel® Core™ i5-6600K processor with four logical cores. The results depicted in Figure 17 show that, on average, 35000 ECDSA keys and corresponding hashes can be generated per second when using four parallel threads. Generating a node ID to enter bucket number 239 (the smallest bucket) would therefore only take 7 s on average. For the real-world implementation of the attack, we pre-computed $5 \cdot 10^6$ node IDs to hijack the lookup-buffer. Computing this many IDs takes roughly 3 min using four parallel threads and 11 min with a single thread. Note that even when attacking different victim nodes this computation has to be performed only once, since the target of a lookup does not depend on any victim-specific information.

3.7.2 *Attack Implementation*

First, to compare the performance of the attack to [168], we repeatedly attacked a recently restarted victim. Our attack relies on connection slots becoming available due to high peer churn. The reliance on churn implies a delay, as previously established connections must be terminated before their slots can be occupied by an adversary. Note that we do not necessarily require a restart of the victim, however, the waiting time is significantly smaller (in the order of minutes) for recently restarted nodes.

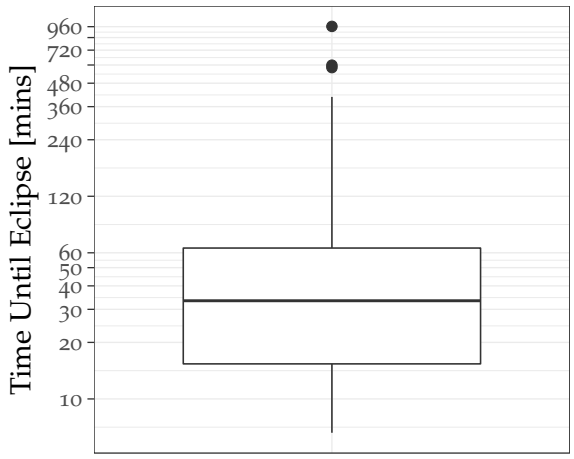


Figure 18: Log-scale box plot of attack durations when attacking a recently started victim. The plot depicts the median duration as well as the upper and lower quartile and all outliers outside 1.5 times the interquartile range.

The attack was started immediately after the victim node became online. No connection slot was yet occupied, but the neighbor table always contained benign nodes, due to countermeasures introduced in [168]. We measured the time until every slot was filled with an attacker-controlled node, with a cutoff timeout of 24 h after which the experiment was restarted. One could argue that also in unsuccessful attempts, the victim would eventually have been eclipsed. The attack was repeated 50 times, out of which 45 times were successful within the cutoff timeout, whereas 5 attempts did not complete in that time. Figure 18 shows the results in a log-scale box plot. The box depicts the upper and lower quartile of measured durations, the median is indicated as a solid line inside the box. Measurements outside 1.5 times the interquartile range are considered as outliers, plotted as dots.

It can be seen that out of the 45 successful attacks 75 % completed in just over 60 min, indicating that an adversary can eclipse recently restarted nodes within a reasonable time span. This finding implies that peer churn in the neighbor table and in the peer management is high in recently restarted nodes.

3.7.3 *Distribution of Connection Durations*

Since recently restarted nodes exhibit low-duration connections and high churn, the question arises if the same behavior is true for long-running nodes, i.e., how connection durations are distributed. To this end, we ran a unaltered Geth node for 450 hours (almost 19 d) and logged the duration of every connection. The results are depicted in Figure 19, showing the cumulative distribution of durations in

the trace. To improve the readability of the figure, we excluded connections with a duration shorter than 60 s, which were 90.3 % of all connections, yielding a total of 361 connections. We suspect the heterogeneity of the discovery table as the major cause for the abundance of connections shorter than 60 s [143]. Ethereum is embedded in a family of protocols (Section 3.3), all of which use the same ports and message structures. Kim et al. [143] report that only 10 % of node IDs in the discovery table correspond to peers speaking the Ethereum protocol, out of which only 50 % operate on the Ethereum main network. Hence, in the majority of situations DEVp2p tries to establish connections to peers which are not useful and immediately discards them again.

It can be seen in Figure 19 that the longest connection duration was 17.4 d, but the majority of connections was much shorter lived. The quantile shows that 95 % of the considered connections were shorter than 5.5 d; only 18 connections were longer than this duration. Though the peer churn is lower than in a restarted node, it is still surprisingly high for which we have several conjectures: Geth's development cycle is fast and requires frequent updates, either due to security fixes or protocol changes. Additionally, read (write) timeouts of 20 s (30 s) on the TCP-level are relatively small compared to other networks like Bitcoin. On the UDP-level, timeouts are set at 500 ms, while [113] report average inter-node latencies of roughly 180 ms, with 10 % of peers having a latency higher than 276 ms. Consequently, buckets in the discovery table experience a high level of churn, making it easy to enter even fully filled ones.

Given that most connections on the Ethereum network are rather short-lived, we conducted a proof-of-concept attack without restarting the victim. We let the victim node run without any attack activity for 72 hours to populate the discovery table, mimicking a more realistic network state. In our experiment the false friends attack was successful after 4.7 d (114 hours). In two subsequent experiments, the victim was left with only one benign connection after 4.9 d and 9.5 d, respectively (i.e., the node was only one connection away from being fully eclipsed). We deliberately did not investigate this further as (1) assuming a recently restarted victim is common for eclipse attacks [131, 168] and (2) restarts happen regularly on the Ethereum network due to the rapid development and subsequent node upgrades.

3.7.4 *Scaling the Attack*

In the description thus far we focused on attacking a single Ethereum node, the attack can easily be scaled to multiple nodes simultaneously. The pre-computation of a large of number of node IDs has to be carried out only *once* as the resulting database of ECDSA key pairs can readily be used for another victim. An attacker also needs only two IPs from distinct /24 subnets to attack an arbitrary number of

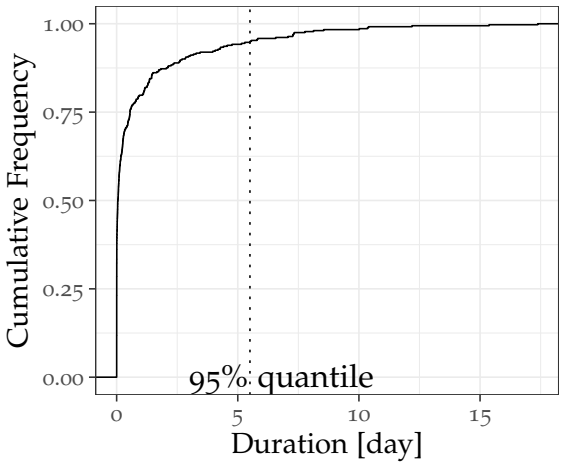


Figure 19: Cumulative distribution of durations in the trace. of the considered durations were shorter than 5.5 d.

nodes, as the restrictions are enforced per node. The main bottleneck when scaling the attack is, therefore, starting a sufficient number of Geth clients.

Recall that connections are established based on (1) a node's buckets and (2) the result of a lookup operation (cf. Section 3.5), i.e., not every inserted node ID is necessarily chosen for a connection. Each started Geth client is equipped with one of the suitably crafted node IDs, fitting the corresponding entry the victim's routing table or respective lookup operation. Therefore, the re-usability of these Geth instances depends on (1) the xor distance of victim node IDs and (2) the random lookup targets.

Consider the case of two victim nodes N_1, N_2 with IDs X_1 and X_2 as an example: when X_1 and X_2 differ in the first bit, the Geth clients with IDs used to populate the buckets of N_1 can only be used to populate the first two buckets of N_2 (and vice versa), as the employed node IDs for N_1 successively share more bits with X_1 the higher their target bucket becomes. Regarding the lookup, since the targets are chosen uniformly at random, they differ already in the first bit with probability $\frac{1}{2}$, making it unlikely that node IDs (and thus Geth instances) can be re-used for another victim node.

In the worst case scenario of maximal different node IDs and lookup targets, 48 Geth clients would be necessary to eclipse all 25 connections of both peers, requiring a moderately powerful server to host these instances of Geth[49]. Hence, as an approximation, the computing resources required to scale the attack rise linearly in the number of victims.

[49] *In our experiments we deployed a server with 16 cores and* 16 GB *RAM to host 25 Geth clients.*

3.8 COUNTERMEASURES

3.8.1 *Proposed Countermeasures*

Before addressing potential countermeasures, let us recapitulate the problems first which lead to the vulnerabilities addressed in the previous sections.

First, node IDs are public and relatively static which, in itself, is not problematic. Although overlays with dynamic node IDs are imaginable (e.g., in the I2P anonymity overlay [89, 278]), static node IDs are a reasonable design choice. However, public and static node IDs are problematic when the entire internal structure of known peers depends *only* on these IDs. The possibility to forecast the bucket structure of any node given its ID (implying the ability to determine which nodes are queried during a lookup) is the major driving factor of our attack on Ethereum as well as other related attacks on Kademlia (cf. Section 3.2).

Secondly, Kademlia exhibits logarithmic state complexity, i.e., in a network with N nodes and bucket size b, each node stores $O(b \cdot \log_b(N))$ routing table entries [247]. While desirable for content distribution, this behavior severely limits the set of nodes to which overlay connections can possibly established. As highlighted by Marcus et al. [168], even when contacting $N = 25.000$ nodes (which was the entirety of the Ethereum network in 2017), only 168 routing table entires would be in the routing table, on expectation. This entails an unnecessarily reduced diversity in the overlay construction process, which is vital for robustness (cf. Section 2.4.2).

Several quickly realizable modifications to Geth are conceivable that will immediately increase the costs of a successful false friends eclipse. As we saw in Section 3.6.2, increasing the bucket size is not promising. One possibility are more stringent *IP subnet restrictions* in the discovery table and on the DEVp2p connection layer. Enforcing *any* subnet restriction on the replies of `FindNode`-Requests would increase the number of unique IP addresses necessary for a successful attack. Another low-invasive countermeasure is to consider all known nodes in `ReadRandomNodes`, instead of only the heads of each bucket. Furthermore, we are only able to perform our attack without requiring a restart of the victim node because peering relationships in Ethereum are currently very short-lived. Increasing timeouts on both the TCP- and UDP-level could decrease this volatility.

On a more fundamental level, we argue that the *complexity* of Geth's current node selection logic is a major enabler for attacks such as [168] and our false friends attack. New peers are chosen based on their node ID, which arguably does not make any sense if the goal of the resulting overlay is flooding identical information to all nodes (in contrast to ID-based routing). Node IDs are, however, trivially manipulatable by adversaries to optimize the placement of adversary

nodes in peer discovery tables. The complexities of ID-based peer selection are therefore not only unnecessary, but also detrimental to security. For a sustainable, long-term fix we strongly suggest to ignore node IDs for all aspects of peer discovery for the Ethereum protocol. Instead, peering decisions should be weighted by more expensive-to-manipulate node characteristics, such as IP addresses or, perhaps, publicly locked Ether stake linked to individual nodes. With respect to minimizing the risk of peer sets filled with adversarial nodes, we suspect that a node selection that closely resembles a uniform draw from the population of all nodes is optimal.

It remains unclear why Ethereum chose a structured Kademlia overlay as the basis for peer discovery, when Kademlia was designed for a completely different use case (cf. Section 2.2). Furthermore, especially with Bitcoin as a prime example of a dependable network layer, specifically designed for the "blockchain scenario", the design choice to go for Kademlia instead of simply copying Bitcoin's well-designed system is not reasonable.

Decades of research on Sybil-attacks in peer-to-peer networks [86] suggest that in a completely trustless setting it is only viable to make the creation of a multitude of adversarial nodes expensive, not impossible. The implications of this are twofold. First, in the typical blockchain scenario one honest node is sufficient to prevent an eclipse attack[50]. Therefore, the probability of filling the whole peer list with adversarial nodes must be minimized by means that are robust to a potentially substantial population of Sybil nodes. Second, nodes that are profitable targets for eclipse attacks (high-profile merchants, miners) should not rely on a purely trustless node selection logic. Instead, these nodes should statically include known and trusted nodes into their peer list, as seems to be practice in the Bitcoin network [131]. In other words, potentially attractive targets might want to invest manual effort to *choose their friends wisely*.

[50] *At least in the case of PoW, it differs for PoS.*

3.8.2 *Implemented Countermeasures*

In response to our responsible disclosure and subsequent discussions[51] the Ethereum developers implemented several low-key countermeasures that have been incorporated into the v1.9.0-release of Geth. These changes mitigate the immediate threat of false friends eclipse attacks and raise the bar for an attacker by requiring an increased number of Sybil nodes to carry out an attack. Admittedly, we still strongly suggest to fade out the structured Kademlia-based discovery for a less structured approach, e.g., the addrman used in Bitcoin.

[51] *We want to thank Felix Lange for his time and the fruitful discussions.*

3.8.2.1 *Raising the Number of Connections to 50*

Before Geth v1.9.0, each node established a total of 25 connections by default, 8 of which are outbound and 17 are inbound. The number of outbound connections in Geth is defined as

$$\#\text{ of outbound} = \lfloor \frac{\text{max peers}}{3} \rfloor = \lfloor \frac{25}{3} \rfloor = 8. \tag{30}$$

The total number of connections a Geth node establishes has been doubled from 25 to 50, effectively doubling the number of outbound connections as well. Therefore, an adversary would require more resources to successfully eclipse a victim. Furthermore, since the false friends eclipse is only successful when existing connections to honest nodes are dropped, the increased limit raises the chances of maintaining a long-lived connection to an honest node, thwarting the eclipse as long as these connections are sustained.

3.8.2.2 `ReadRandomNodes` *Considers All Nodes*

The function `ReadRandomNodes` now selects nodes uniformly at random from the set of all nodes in the table, instead of just the bucket heads. An adversary would have to overtake the whole table to deterministically ensure that only adversarial nodes are returned. However, the maximum number of nodes in the table is relatively small: 17 buckets with a maximum of 16 nodes, i.e., 272 possible nodes in total. Assuming 18 adversarial nodes as used throughout this paper, the chances of selection an attacker-node at random are $\frac{18}{272} \approx 7\,\%$. In practice, having a full table is very unlikely, due to Kademlia's distance metric: the size of the potential node set of each bucket decreases exponentially. While the first buckets are always filled, the later ones tend to be almost empty most of the time. Marcus et al. [168] report an average table population of 168 nodes, therefore increasing the chances of randomly selecting an adversarial node to $\approx 11\,\%$.

We argue that while selecting peers uniformly at random is a desirable strategy with respect to robustness, the node set from which peers are drawn should be sufficiently large.

3.8.2.3 *Throttle Inbound Connection Attempts*

A major facilitator of our false friends eclipse is the ability to establish inbound connections from the same IP address. All inbound connection slots could be filled during our evaluation by running just one server. Since IP addresses are the most costly part in such an attack (in comparison to memory or computational power), increasing the necessary number of addresses to fill inbound connection slots is vital to raise the bar for an attacker.

Since Geth v1.9.0, inbound connection attempts from the same IP now have to wait 30 s. While this is a first step in raising the bar

for an attacker, we argue that this is not enough. Additional subnet restrictions on inbound connections (e.g., only 2 IPs from the same /24 subnet) are an effective and low-invasive way of making an eclipse more difficult.

3.8.2.4 *Unchanged: The lookup-buffer*

A major component of our false friends attack is the exploitation of the lookup buffer (cf. Section 3.5.2). With one adversarial node in each bucket (which is still possible) and a sufficiently large number of pre-computed node IDs, an adversary can still ensure that the lookup-buffer is filled with only adversarial nodes. Since `ReadRandomNodes` cannot be exploited without a significant resource-investment, a compromised lookup-buffer does not pose an immediate threat but implies a non-negligible probability of choosing an adversarial node from the discovery table with every call of `ReadRandomNodes`. Hence, the threat of conducting an attack similar to ours is not convincingly mitigated. We strongly suggest to enforce subnet restrictions or ignoring the lookup-buffer completely for peer selection [52].

[52] *The lookup-buffer and further mitigations have potentially been addressed in later releases of Geth.*

3.9 ECLIPSE ATTACKS IN THE WILD

Given the potential attack vectors laid out in the previous sections, how realistic is the threat of eclipse attacks on blockchain systems such as Bitcoin or Ethereum[53]? Although there have been a plethora of (partially successful) attacks on Bitcoin, Ethereum and the like, to the best of our knowledge, there has been no reported case of adversarial eclipse attacks on Bitcoin and Ethereum or any other major cryptocurrency. While surprising initially, there are reasonable economic arguments in terms of the cost/benefit ratio why eclipse attacks are unpopular amongst adversarial persons: eclipse attacks are complex and induce costs that are not matched by the potential profit in comparison to other attacks. We will elaborate on the potential costs and benefits in the following.

[53] *Apparently large enough to receive a corresponding bug bounty.*

First, *targeted* eclipse attacks are hard to launch, due to the information required. While it is easy to simply launch an attack as described in this chapter on a random node, eclipsing a specific target to steal funds through, e.g., double spending, is a significantly harder challenge. When identifying a potential target an adversary has to gain knowledge of the target's network layer address by only having information about its addresses (Bitcoin) or accounts (Ethereum). Both Bitcoin and Ethereum provide, for a good reason, no link between coins and peers which hold them — Bitcoin even tries to obfuscate the origin of transactions further through diffusion (cf. Section 2.4). Hence, an adversary needs at least some knowledge on the network topology to establish the link between coin holder and node on the

network. This is not easily obtainable and requires large-scale measurements [207].

Second, eclipse attacks require a constant investment of resources. On the one hand, maintaining the attack itself by running the necessary nodes needs time and resources, but apart from a Denial-of-Service (DoS) not much is gained through simply eclipsing a peer. Double spending funds or obtaining a mining advantage, the most commonly cited "use cases" for eclipse attacks [131, 168, 205], require substantial amounts of mining power. Therefore, even when eclipse attacks are possible with little time and resources, as demonstrated in this chapter, turning that attack into profit involves further investments.

In contrast to Ethereum, attacks on Bitcoin require substantial resources, for example. Heilman et al. [131] were the first to showcase eclipse attacks against Bitcoin — attacks which require a significant number of distinct IP addresses and can therefore "not easily be launched from a legitimate cloud service", but require the resources of a small botnet or organization instead [131]. Further hardening of Bitcoin since the first eclipse attack's disclosure has additional raised the bar for eclipse attacks [255]; requiring 5-6 weeks of Tier-1 or large Tier-2 ISP resources to divert BGP traffic and inject packets in real time. Furthermore, this assumes no errors or setbacks during this period, rendering the eclipse attack presented in [255] fully impractical and easily solvable by encrypting Bitcoin's traffic[54].

[54] *However, as demonstrated in [15], BGP-based routing attacks on Bitcoin can severely degrade the performance of the network*

Third, other forms of attacks yield higher rewards with higher certainty. Given the difficulty in launching targeted eclipse attacks, resulting in uncertainty about the potential benefit, the amount of gains through a mining advantage or double spends are small in comparison to other attacks. Especially for Ethereum, funds can be stolen through the exploitation of smart contracts, as has been demonstrated vividly in the past [19, 60, 66, 214, 223, 226, 282] — a process which is more reliable and requires less resources than an eclipse attack. Due to the rapidly emerging technology, new attack targets emerge, as has been shown by Gudgeon et al. [126] and Qin et al. [222] in the context of decentralized finance (DeFi). Decentralized finance is an architecture on top of blockchain platforms (e.g., Ethereum), consisting of decentralized exchanges, lending markets and derivatives [126] — allowing for a variety of new attacks and possibilities to steal funds. But also simply scamming people for their wallet's private keys or launching 51 % attacks on altcoins with low total hash rate are viable, probably easier and more failsafe ways to steal money instead of eclipse attacks.

However, one should not be fooled into a false sense of security. Although *targeted* eclipse attacks are probably unlikely to occur, eclipsing random nodes on the Ethereum network requires little resources, as we have demonstrated in this chapter. The victim still experiences as DoS which can, at a larger scale, even threaten the performance of the network as a whole.

3.10 CHAPTER SUMMARY

3.10.1 *Summary*

In this chapter we have seen that eclipse attacks on Ethereum are possible. To this end, we presented the *false friends* attack, an eclipse attack applicable to Geth ($\leqslant$ v1.9.0), the by far most popular Ethereum node software. Our attack requires little resources: only 2 IPs from distinct /24 subnets are sufficient for a successful attack. Moreover, and in contrast to previous attacks, it can principally be successfully mounted without assuming that the victim node reboots at some point. Our discovery is even more striking when considering that countermeasures against similar attacks were only recently introduced to the Geth codebase and did not pose a significant challenge to circumvent. We argue that the ongoing vulnerability of Geth is due to a fundamentally unsuited node discovery approach. While we propose both short- and long-term countermeasures to the false friends attack, existing literature hints that in a completely trustless setting, eclipse attacks can only be made expensive, not impossible. Potentially attractive targets might wish to invest manual effort towards *choosing their friends wisely* by establishing trusted connections to other known nodes on the network, e.g., through interconnections between exchanges and/or miners.

3.10.2 *Conclusion*

Linking back to the introductory remarks and questions from Chapter 1: are ascriptions of robustness towards Ethereum appropriate?

On the one hand, the high degree of replication as well as the sheer size of the Ethereum network makes it unlikely for an eclipsing adversary to significantly impact the network as a whole. While individual nodes may fall victim to eclipse and subsequent DoS attacks, a network-wide partition is, at least with the attack vectors presented in this chapter, unlikely. In this regard, due to its decentralized nature, Ethereum can still be considered robust.

On the other hand, our proof-of-concept eclipse attack is a clear argument against the narratives of robustness and instead reveals the fragility of Ethereum's networking logic. Although targeted eclipse attacks to double spend funds or exclude specific network participants from the remainder of the network are improbable, non-targeted DoS attacks still remain an evident issue. Especially due to the low amount of necessary resources, simply "trying out" becomes a viable option for an adversary, which is also why we chose not to publish our code of the attack. Due to the permissionless and "trustless" setting, the problem of eclipse attacks is unlikely to be solved by purely technical means, which is why we suggest to utilize existing trust relationships

between high-value actors. From this perspective, Ethereum cannot be considered robust, especially since manual intervention with regards to overlay connections is advised.

In the next chapter we will examine IPFS, a popular system for distributed data sharing, as a representative of the storage layer in our blockchain stack model (cf. Section 2.6). Similar to Ethereum, IPFS is also based on a Kademlia-style DHT, suggesting the possibility of similar attack vectors. As we will see, IPFS employs a hybrid overlay of structured Kademlia lookups combined with unstructured flooding to one-hop neighbors. This unusual combination and the high number of established connections renders IPFS significantly more robust against eclipse attacks and effectively thwarts the attack vectors presented in this chapter. However, while increasing robustness, IPFS purposefully sacrifices performance and, as we will see in the next chapter, user's privacy through this approach.

4

MAPPING THE INTERPLANETARY FILESYSTEM

4.1 OVERVIEW

Decentralized, peer-to-peer-based data storage systems are becoming increasingly popular, especially in the context of blockchain applications and censorship-resistant data hosting. As laid out in the beginning of this thesis (cf. Chapters 1 and 2), narratives around previously conceived data storage systems such as BitTorrent and related filesharing networks [217, 263] were mainly focused on performance and scalability, whereas newer projects put a stronger emphasis on resilience against attacks and censorship. This shift in narratives is also reflected in the systems' designs, e.g., in an increased usage of techniques common to unstructured overlay networks (as in Bitcoin, cf. Section 2.4). The Interplanetary Filesystem (IPFS) is a prominent example of a newer P2P data storage system [34] — a community-developed peer-to-peer protocol and network providing public data storage services. IPFS employs a hybrid approach between a structured Kademlia overlay and broadcasting of so-called BitSwap requests for content to directly connected peers, further supporting the hypothesis of shifted narratives and system designs.

As a data storage system, IPFS is often cited as a fitting solution for blockchain-based applications [13, 17, 159, 160, 201, 209, 234, 253] and was previously used for mirroring censorship-threatened websites such as Wikipedia[55]. It is therefore located on the storage layer of our blockchain application model (cf. Section 2.6), serving as a prime example of distributed storage systems. Any Internet-enabled device can participate as an IPFS node and nodes are operated without explicit economic incentives. Data items (files, folders, ...) are not replicated globally but are instead hosted by a small set of nodes, that item's *providers*, who make the data item available to other peers. Data

[55] *https://github.com/-ipfs/distributed–wikipedia-mirror*

This chapter is based on extensive previous collaborative work [1, 6, 7]. The idea for crawling the IPFS network, understanding its structure, as well as the initial crawler design and evaluation should be attributed to myself (cf. Sections 4.3 to 4.8) — in close coordination with Martin Florian. Sebastian Rust joined in the process and significantly improved the quality of this work by redesigning the crawler and through insightful discussions.
The idea of monitoring BitSwap requests originated in a discussion with Martin Florian, was validated by myself in a proof-of-concept and subsequently became the topic of Leonhard Balduf's master's thesis which I closely supervised. As a rule of thumb: all heavy lifting on the data-side has been conducted by Leonhard Balduf, my very own contributions are, besides the close supervision and rigorous discussions, the popularity distributions (cf. Section 4.10.5) & the estimation of network size, based on an idea of Martin Florian (cf. Section 4.9.3).

items are addressed through immutable, cryptographically-generated names which are resolved to their providers through a distributed hash table based on Kademlia (cf. Section 2.2). While other solutions exists, e.g., Ethereum's Swarm [106] and BitTorrentFS [39], IPFS can be considered to be the maturest and most widely used system.

Given IPFS' reported attractiveness as a building block for decentralized applications and censorship circumvention, the question arises whether the network is actually suited to fulfill this role in terms of robustness and decentralization. We are particularly interested in the following questions:

- What are possible (non-exhaustive) avenues for monitoring and mapping the IPFS network and what limitations do they entail?
- How many and what types of nodes participate in the IPFS network? What kind of churn do they exhibit?
- How "decentralized" is the network — for example, in terms of overlay structure and geographical distribution of nodes?
- How secure/robust is the current design against adversarial nodes, especially regarding its open nature and the inherent Sybil problem (cf. Section 2.4.1)?

In this chapter, we present the results of comprehensive empirical studies on the IPFS overlay network to address these questions[56]. At the heart of our methodology lies the exploitation of IPFS's aforementioned hybrid architecture for content retrieval: (1) Kademlia DHT lookups combined with (2) broadcasting of data requests to all connected neighbors. In particular, we investigate possible avenues for measurements and (potentially unintended) monitoring enabled through this design.

With respect to (1), the DHT, we find that similar to other Kademlia-based systems [217, 246], connections corresponding to DHT routing table entries can be learned through carefully crafted, iterative peer discovery queries. To this end, we developed ipfs_crawler, specifically designed to extract IPFS's overlay topology[57]. We crawled the IPFS DHT back-to-back, i.e., starting the next crawl as soon as the previous finished, for 7 d in Nov. 2019 (C_1) and again in Feb. 2021 (C_2) for a period of 14 days — results relate to the Nov. 2019 crawl, with comparisons to Feb. 2021 when appropriate (the complete report for C_2 can be found in the Appendix, cf. Section 7.1.1). The obtained dataset enables insights in the geographic distribution of nodes, graph properties of the overlay and usage patterns. For example, we suspect that most nodes are operated by private individuals on a "as needed" basis. Furthermore, in contrast to prior works on Kademlia [232], IPFS' overlay network does not seem to be scale-free, especially in the case of graphs from C_2, whose degree distributions do not plausibly stem from a power-law, log-normal or Poisson distribution. Interestingly,

[56] *A condensed version of these results can be found in my talk at ProtocolLabs, https://www.youtube.com/watch?v=-jQI37Y25jwk.*

[57] *The code of our ipfs_crawler and evaluation is maintained at* `https://github.com/wiberlin/ipfs-crawler`. *We use ipfs_crawler to conduct and visualize periodic measurements at https://trudi.weizenbaum-institut.de/ipfs_analysis.html.*

this renders the IPFS overlay robust against random failures and targeted removal of nodes with the highest degree.

While crawling the DHT enables insights into the overlay topology, (2) the broadcasting data requests, enables the monitoring of data access patterns and trends. We present a passive monitoring methodology for collecting and processing BitSwap data requests of a large share of the network and a monitoring setup as an instance of the methodology[58]. Our system enables us to reveal *who* requested *which* data item *when*, i.e., which nodeID and IP address requested which content identifier (CID) at what timestamp — putting the privacy of users at risk. We collected measurements for nine months using two spatially diverse monitoring nodes, yielding traces of 9.68×10^9 data request entries in total.

[58] *The corresponding tools are maintained at https://github.com/-mrdoll4r/ipfs–resolver.*

Equipped with the obtained dataset, we highlight possible angles for analyses:

- the estimation of the size of the network (including non-DHT nodes),
- analyses of activity levels and structure (e.g., geography-based usage patterns),
- derived content popularity distributions, and
- the feasibility of privacy attacks, by identifying node IDs of HTTP/IPFS gateways.

An adversary with a similar setup to ours can determine (1) which IP addresses are interested in a given CID, (2) which CIDs were requested by a particular node, and, with negligible deniability, (3) whether a node (and hence its user) has downloaded a specific (CID-referenced) data item in the recent past.

The remainder of this chapter is structured as follows. After discussing related work in Section 4.2, we give a concise overview on the IPFS system based on white papers, public online discussions, and code in Section 4.3. Notably, we describe the state of IPFS v0.4[59] implementations, and contrast it with the design documents, when necessary. Commencing with the DHT aspects, in Section 4.4 we describe the limits of what can be learned through crawling. Subsequently, we present the results of a ground-truth setup in Section 4.5, allowing us to quantify the limits described in Section 4.4. In Section 4.6 we first describe our ipfs_crawler which we developed specifically for the task of obtaining snapshots from the IPFS DHT. The results of the crawl are described in Section 4.7 for which we repeatedly crawled the network to obtain its overlay topology, thereby also enumerating all DHT-enabled nodes and their addresses. Equipped with the data of these crawls, we analyze usage patterns, spatial distribution of nodes and graph properties of the overlay. After an interim conclusion (cf. Section 4.8), we move on to BitSwap, introducing our monitoring

[59] *With additional notes to subsequent versions v0.5 and v0.6 which incorporated DHT hardening efforts in reaction to Eclipse attacks [220] as well as significant implementation changes to the DHT in general.*

methodology and setup in Section 4.9. Specifically, we showcase methods to estimate the number of nodes in the IPFS network based on combining measurements of several monitoring nodes, as well as definitions on content popularity. Subsequently, we apply the proposed methods to our BitSwap dataset in Section 4.10. Last but not least, we end this chapter with a discussion of privacy risks implied by our methods and potential privacy-enhancement approaches (Section 4.11) as well as concluding remarks in Section 4.12.

4.2 RELATED WORK

A variety of P2P systems have been measured and monitored in the past, through passive measurements, active crawling and probing, as well as combined approaches. This vast body of literature provides various fundamental insights on network crawling and characterization [141, 176, 233, 243, 244, 246, 248, 249, 277], with applications to real-world P2P systems like Gnutella [128, 248, 251], BitTorrent [217, 263] and KAD [176, 244]. In the following we will spotlight a selection of important insights.

In their seminal work, Stutzbach and Rejaie [248, 249] study requirements and pitfalls with regards to obtaining accurate snapshots of P2P overlays. Specifically, the duration of crawls should be as small as possible, to avoid distortions in the results due to churn. Steiner et al. [243, 244] crawled the KAD network to obtain the number of peers and their geographical distribution as well as inter-session times. Wang and Kangasharju [263], in response to [243], highlight the importance of using more than one geographic vantage point and discuss important considerations when combining network size estimates from crawls. Salah et al. [232] studied the graph-theoretical properties of KAD and contrasted their results with analytical considerations, indicating that resilience to random outages as well as degree distributions coincide with simulations. Similarly to KAD and other networks, the DHT component of IPFS is also based on Kademlia [170].

More recently, P2P networks have received renewed attention in the context of cryptocurrencies such as Bitcoin and Ethereum [257]. In a measurement setup similar to our BitSwap monitoring (cf. Section 4.9), Neudecker et al. [207] inferred the topology of the Bitcoin overlay through monitoring block and transaction distributions from several spatially diverse nodes. A similar approach is repeated in [33]. Network-level measurements for topology inference have also been conducted for the privacy-focused cryptocurrencies Monero [49] and ZCash [69]. Apart from topology inference, the latency and bandwidth of Bitcoin and Ethereum peers was measured to assess the systems' degree of centralization [113] and to assess the network health of Ethereum in general [111, 143], indicating centralization tendencies as well as massive inefficiencies in Ethereum.

We extend this line of research on P2P systems by developing and performing measurement studies on IPFS — a highly popular data storage network with various reported applications (see, e.g., [159, 160, 253] for a sample of academic projects). Although becoming increasingly popular, IPFS has so far not been studied extensively by the scientific community. Mostly, the I/O performance of retrieving and storing data was studied in the past [17, 61, 62, 237], with partially contrasting results. Furthermore, the end-to-end performance was studied at the example case of DTube, a decentralized replica of YouTube hosted on IPFS [85]. Ascigil et al. [17] report high latencies, low throughput and high redundancy when retrieving data through IPFS. Similarly, Shen et al. [237] report high latencies for large files and large variances in the transmission speed. In [61, 62], the authors optimize and evaluate the performance of IPFS in edge computing settings. They report small latencies and high throughput when using the global DHT as little as possible and running IPFS in a private local network.

In contrast to these prior works on IPFS, we focus on grasping the overlay structure and node composition of the public IPFS network. Furthermore, we give a comprehensive, code review-supported overview of IPFS' "network layer", revealing information not previously available in literature or documentation. We find that clients maintain a large number of connections, a subset of which is stored in the k-buckets of the underlying DHT and can be assessed through crawling, which is covered in the first half of this chapter. Requests for data are broadcast to *all* immediate neighbors, a DHT search is secondary and only performed if none of the neighbors have the data. While the "unstructuredness" helps the network to be more resilient against attacks, e.g., eclipse attacks [238], to some extend, it comes at the cost of privacy. This insight forms the basis of our following investigation, as we exploit IPFS' broadcasting behavior through passively collecting request messages from peers to learn about data shared on IPFS — the second half of this chapter.

We build upon previously proposed crawling and measurement methods (e.g., [246, 248, 263]). However, we also find that IPFS differs substantially from more canonical Kademlia implementations, necessitating enhancements to existing measurement approaches, especially with regards to crawling. A simple crawler for the IPFS DHT has been made available before[60] that aims at enumerating all nodes in the network. For this work, we developed a new crawler from scratch to capture the entire overlay topology. It is optimized for short running times to ensure the correctness of snapshots.

[60] https://github.com/vyzo/ipfs-crawl

Last but not least, although resilient, DHT is not invulnerable to attacks. As showcased by Prünster et al. [220], IPFS' connection manager could be gamed to eclipse a peer with high probability — including its non-DHT connections. Since IPFS v0.5, several countermeasures

have been introduced to mitigate the attack, increasing the necessary resources by several orders of magnitude [220]. Also, due to IPFS's unstructured broadcasting of queries, eclipse attacks on IPFS are significantly harder than in other Kademlia-based networks, e.g., Ethereum (cf. Chapter 3).

4.3 THE INTERPLANETARY FILESYSTEM

In the following, we describe key aspects of IPFS' design and discuss particularities relevant to conducting measurement studies and interpreting their results. It is worth noting that the development of IPFS is ongoing, so that details of the design may change over time. Here, we focus on inherent conceptual properties that change rarely in deployed protocols.

4.3.1 *In a Nutshell*

As a broad overview, the design of IPFS can be summarized in the following way:

- IPFS is a permissionless system with weak identities; anyone can deploy a node on the IPFS overlay network.
- Data is transformed into a directed acyclic graph structure (so-called Merkle DAG); each node is identified by the hash of its content.
- Data items are stored and served by data providers, references to which are stored in a Kademlia-based DHT
- Data items are requested from *all* connected overlay neighbors and the DHT is queried for providers only *after* no neighbors were able to offer the data.
- In contrast to information in the IPFS white paper [34], no proposals of S/Kademlia [29] are implemented.
- Overlay connections can correspond to DHT routing table (bucket) entries, but do not have to.
- By crawling the DHT we obtain a subset of all connections; we estimate the size of that subset in Section 4.5.2.
- BitSwap resembles BitTorrent [217] Bitcoin's inventory mechanism [257]; by monitoring all incoming messages we obtain a trace of who requested what data when (with some limitations).
- It is a default behavior for nodes that have downloaded a given data item to *cache* it locally, effectively becoming a data provider for that item.

4.3.2 *Node Identities and S/Kademlia*

Anyone can join the IPFS overlay network, i. e., it is an open (permissionless) system with weak identities. Nodes are identified by the hash of their public key, $H(k_{pub})$. To ensure flexibility, IPFS uses so-called "multi-hashes": a container format capable of supporting different hash functions. A multi-hash adds meta information about the hash function and the digest length to a hash. By default, IPFS uses RSA2048 key pairs and SHA256 hashes.

Creating a new IPFS identity is as simple as generating a new RSA key pair — making the network highly susceptible to Sybil attacks [86]. Towards increasing the cost of Sybil attacks, the IPFS white paper [34] suggests that the Proof-of-Work-based ID generation approach of S/Kademlia [29] is in use. However, based on our careful review of the IPFS codebase, this is *not* the case. IPFS versions $<$ v0.7 implement no restriction on the generation of node IDs, neither are DHT lookups carried out through multiple disjoint paths, as proposed in S/Kademlia. IP address-based Sybil protection measures, such as limiting the number of connections to nodes from the same /24 subnet, are also *not* in use.

On the network layer, IPFS uses a concept of so-called "multi-addresses" (or multiaddr for short). Similar to multi-hashes, these multi-addresses are capable of encoding a multitude of network- and transport layer protocols, including nested and proxied constructions. Through multiaddrs, a node announces its capabilities (e.g., IPv4, IPv6, QUIC, ...) and the network layer addresses at which it can be reached.

4.3.3 *Content Identifiers (CIDs) and Data Integrity*

IPFS uses a form of self-certifying filesystem (SFS) [171] to ensure the integrity of data throughout its delivery. To this end, each data item d is assigned a unique immutable address that is the hash of its content, i.e., $\text{addr}(d) = H(d)$. Recipients can recognize whether received data was tampered with by comparing its hash with the requested address. In IPFS, an $\text{addr}(d)$ is encoded as a so called *content identifier* (CID).

Directories and files are organized as a Merkle DAG[61]. This construction differs from Merkle Trees insofar as nodes can have more than one parent and, in the case of IPFS, data on non-leaf nodes is permitted. For example, a directory on IPFS is encoded as a node containing the hashes of all entries in the directory in addition to metadata about each entry. Edges are directed from the upper layers of the directed acyclic graph (DAG) to the lower levels with no possibility to traverse the graph in the other direction. Large files are chunked into smaller data *blocks* and encoded as multi-layered DAGs. This construction ultimately allows for caching and deduplication of both file contents and directory entries.

[61] https://docs.ipfs.io/concepts/merkle-dag/

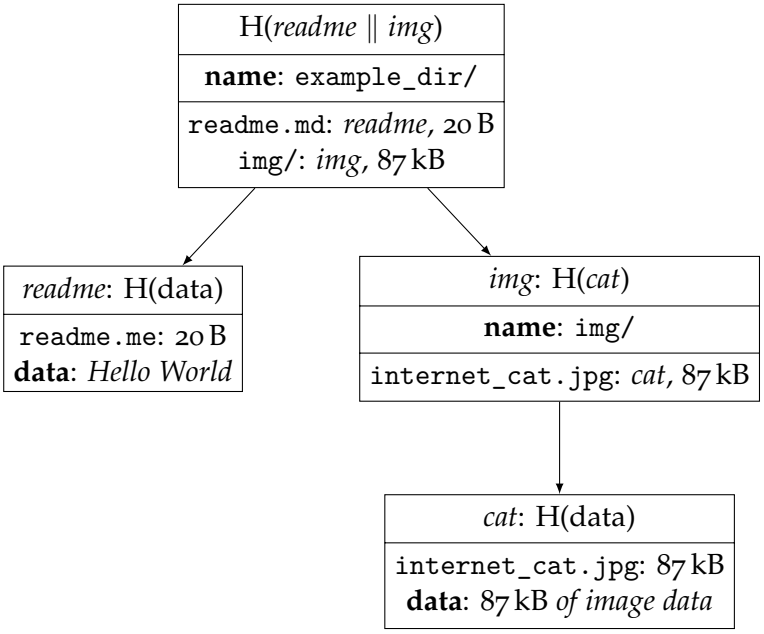


Figure 20: IPLD tree of a simple directory.

In principle, IPFS can be used to store a variety of different contents. The encoding of a data item can be derived from its CID already, using a mapping known as Multicodec (conceptually similar to multiaddresses and multihashes). Important Multicodecs for IPFS are

1. DagProtobuf, which encodes nodes for the IPFS Merkle DAG. These objects usually encode files and directories on IPFS.
2. Raw, which are unencoded chunks of binary data. For IPFS as a file system, these objects usually appear as (leaves of Merkle DAG encoded) files.
3. DagCBOR and DagJSON, which are to-be replacements for DagProtobuf. They encode the Interplanetary Linked Data (IPLD) data model in the respective format. The IPLD data model is a generalized formalization of hash-linked data types.

As an illustrative example, consider the directory in Figure 20, consisting of a subfolder and two files. The folder `example_dir/` consists of a file `readme.md`, as well as a subfolder `img/` which holds the image of the "on the Internet, nobody knows you're a cat" cartoon depicted in Figure 9 in Chapter 2. IPFS partitions data items into blocks, each of which is identified by its respective CID. A CID is simply the multihash (denoted by H(.) in Figure 20) of a block's (meta) data.

[62] *If files are too large, they are additionally chunked into several blocks*

Given such a directory, IPFS starts building the Merkle DAG in a bottom-up fashion. First, blocks are formed containing the files at the leaves of the directory tree, the `internet_cat.jpg` image in our example[62]. The resulting CID of the cat-block is formed by hashing

the image data. Moving upwards, the `img/` directory is modeled as a block with exactly one link to the CID of the cat-block. Directories contain meta data such as their name as well as the names and sizes of the files within them. Note that IPFS leverages this fashion of structuring data for deduplication: if other directories were to contain `internet_cat.jpg`, they would link to the same CID. The CID of `img/` therefore depends on the CID of the cat-block which, in turn, depends on the data contained in `internet_cat.jpg`; yielding the self-certifying property. Lastly, the CID of `example_dir` is computed, as it references the CIDs of the readme- and img-blocks. This topmost CID is also referred to as the *root hash*. Hence, when writing about a data item d, we normally refer to the root hash of d.

4.3.4 *Content Retrieval*

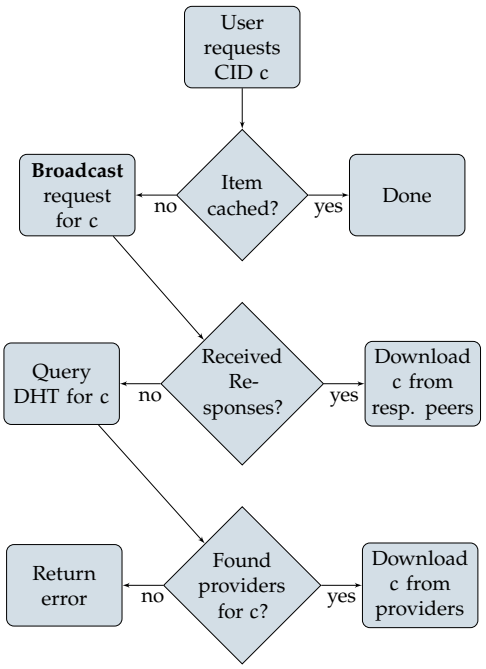


Figure 21: Content lookup in IPFS: a combination of DHT and flooding.

Data items are usually stored at multiple nodes in the network. Nodes store content because they are its original author or because they have recently retrieved it themselves — there is no automated replication of content. Nodes normally serve the data items they store, upon request. The nodes that store a given data item are consequently referred to as that data item's *providers*.

When an IPFS node v wishes to retrieve a data item with CID c (e.g. based on a user request), it follows a two step strategy (s.a. Figure 21):

1. Ask *all* nodes it is currently connected to for c, using the BitSwap subprotocol (Section 4.3.6)
2. If the first step fails, look up the providers P(d) for c in the DHT, then request c from members of P(d) (Section 4.3.5).

Peers discovered through either stage are added to a *session* S(c), which is used to scope subsequent request for data related to c. In the general case, c initially references the root of a DAG of blocks, which v subsequently requests from the peers in the session.

4.3.5 *Content Retrieval: Kademlia DHT*

IPFS uses a Kademlia-based DHT, with $k = 20$ (cf. Section 2.2). Therefore, nodes and data records in the IPFS overlay network are uniquely identified and located by their respective node IDs and CIDs. Records in the IPFS DHT are lists of data providers for data items stored in IPFS, and the keys to these records are consequently the addresses of data items (H(d) for a data item d). An example scenario illustrating this relationship is presented in Figure 22. Nodes can be

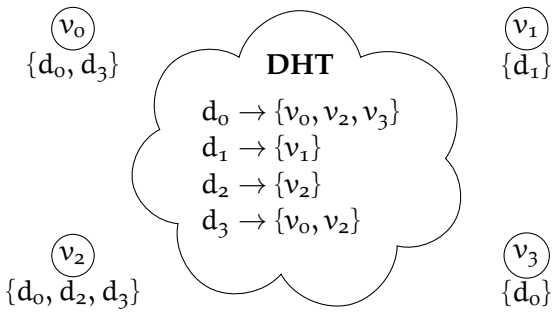


Figure 22: Data items and the DHT.

part of the IPFS overlay network but choose to not participate in the DHT, by acting as so-called *client nodes*. Therefore, we distinguish between the *IPFS overlay* which entails all nodes, including clients, and the *overlay without clients* which consists of nodes that take part in the DHT protocol. When no ambiguity may occur, we simply use the term overlay for both.

IPFS uses a performance optimization in that buckets are *unfolded*, i.e., created, only when necessary. As each bucket corresponds to a common prefix length, the "later" buckets with a longer common prefix length tend to be mostly empty, as the probability to encounter a node that falls into a specific bucket decreases exponentially with the common prefix length. To avoid storing mostly empty buckets, IPFS creates them on the fly when necessary. This in turn implies that we can not know in advance how many buckets a node on the network has currently unfolded. Therefore, we employ a heuristic when crawling a peer: if we receive the same nodes as a response

to two subsequent requests we conclude that the farthest bucket has been reached and cease polling the peer.

Peer-to-peer connections in IPFS are always based on TCP or other reliable transport layer protocols. Consequently, the DHT component of IPFS also uses (TCP-) connections (unlike many other Kademlia implementations, which are datagram based [170]). Nodes are ejected from buckets only if the connection with them is terminated for other reasons and they have been inactive for 10 min, i.e., a ping has not been successful.

4.3.6 *BitSwap*

The BitSwap protocol is the main "data trading module"[63] of IPFS. It is similar to BitTorrent [217] and is used for obtaining data items from connected peers. BitSwap encompasses both (1) announcing interest in CIDs and discovering providers, and (2) actually requesting and receiving the referenced data. For both purposes BitSwap builds upon a reliable transport layer such as TCP, QUIC, or even WebSockets.

[63] https://github.com/ipfs/go-bitswap

4.3.6.1 *Synchronizing Data Requests*

For announcing interest in certain CIDs and discovering providers, so-called `want_lists` are synchronized with connected peers, i.e., each node v maintains a `want_list` for each node it is connected to. A `want_list` for a peer p consists of all CIDs that p is currently looking for (as far as known). To notify relevant peers about desired content, and thereby keep `want_lists` up to date, nodes usually send out BitSwap messages (e.g., collections of `WANT_HAVE` or `WANT_BLOCK` entries as described below) containing changes to previous `want_list` states, which are then applied as deltas. While it is possible to send "full" `want_lists`, our measurement results indicate that this is done rarely. If v receives a wanted block, it broadcasts a BitSwap message containing a `CANCEL` entry to connected nodes; telling them that the block in question is no longer requested. `want_lists` are not persisted when a peer disconnects and re-issued upon reconnection. Additionally, there are mechanisms in IPFS that cause nodes to re-broadcast `want_list` entries in 30 s intervals if the referenced data has not been downloaded yet. These messages serve little purpose for IPFS, since `want_list` entries are kept synchronized and persisted as long as the peers stay connected.

4.3.6.2 *Transmitting Data*

Assuming a user wants to retrieve a data item d referenced by a root hash h, the user's local IPFS node has to traverse the Merkle DAG (cf. Section 4.3.3) downwards in order fulfill the request. To that end, and for each unknown CID encountered during the process,

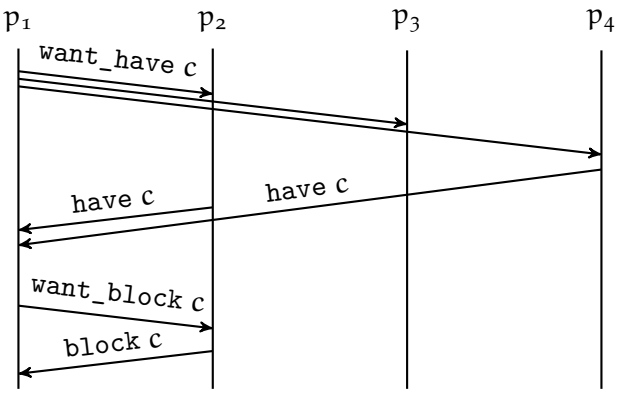


Figure 23: Obtaining a block with CID c via BitSwap.

the node adheres to the flow chart in Figure 21, i.e., it broadcasts a request for the respective block to its neighbors and, in case of no successful responses, starts an iterative lookup in the DHT. In the case of successful responses to its broadcasts, it will cease to broadcast requests for CIDs further down the Merkle DAG and will instead focus on asking the peers which have responded to its initial request. The mechanism is illustrated in Figure 23.

There are fundamentally two types of requests for transmitting actual content data: `WANT_BLOCK` and `WANT_HAVE`, each referencing a single CID. While these requests are, conceptually, messages, they are in practice realized as collections in BitSwap messages supporting multiple entries per message. Since IPFS v0.5 and the introduction of `WANT_HAVE`, the BitSwap data transmission mechanism resembles Bitcoin's use of inventory messages [257]. The `WANT_BLOCK` request type is the backwards-compatible formalization of "I am looking for this block, send it to me if you have it". The `WANT_HAVE` request type was newly introduced and can roughly be understood as "I am looking for this block, do you have it?". Whereas `WANT_BLOCK` triggers the transmission of a block of data (leading to a high redundancy at the receiving side [17]), `WANT_HAVE` only signals the presence of the requested block. Figure 23 depicts a typical procedure for obtaining a data block with CID c using both request types (node p_1 resolves c via its peers p_2, p_3, p_4, with p_2 and p_4 forming a session). Table 1 lists the BitSwap request types used in IPFS v0.5 and above.

Providers of a data item d respond to a `WANT_HAVE` on CID c = addr(d) with a `HAVE` message. The session S(c) for c is formed based on received `HAVE`s and can be complemented with the results P(d) from a DHT search (as noted previously; cf. Figure 21 for a visual overview). `WANT_BLOCK` requests for c are sent to a subset of the peers in S(c). This allows for optimizing requests for *related* content as well as improving request parallelism and reducing redundancy: if a peer has d, it probably also has some child blocks of d, if not all of them. Having established a session for d, future requests for blocks related

Want Type	Cancel	Name	Function
Block	No	`WANT_BLOCK`	request to send a block identified by a given CID.
Have	No	`WANT_HAVE`	request to indicate block availability for a given CID.
*	Yes	`CANCEL`	cancels a previous request of any type for the given CID. Entries remain in the `want_list` indefinitely until either a `CANCEL` is received or the peer is disconnected.

Table 1: Request types of the BitSwap protocol as of IPFS v0.5. Each requests references a CID and is described using the Want Type and Cancel flags. Each response is to be directed at the requester.

to d can be directed at the relevant peers rather than flooded to all connected peers.

4.3.7 *Caching Rules*

One of the keystones of IPFS' design is the caching and reproviding of requested blocks. By default, the IPFS node software stores up to 10 GB of block data, with an optional garbage collection mechanism being activated if the configured storage limit has been exceeded. The garbage collector prioritizes some groups of blocks, such as blocks containing metadata, attempting to keep them in the cache for as long as possible. Users can also *pin* files to ensure their local availability; the garbage collector then traverses the Merkle DAG and marks all blocks associated with the pinned file, henceforth avoiding their removal.

4.4 UNDERSTANDING THE OVERLAY STRUCTURE

IPFS nodes can be separated into two types: nodes that participate in the DHT and nodes that do not, i.e., *clients*. We want to disentangle the different kinds of overlays that arise through this distinction and reason about what information can be measured. We distinguish between the IPFS overlay ($\tilde{G}$), the IPFS overlay without clients (G) and the overlay induced by DHT buckets (G'). We explain the differences in the following.

Overlay networks are commonly modeled as graphs with nodes as vertices and connections between those nodes as edges. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be an undirected graph (due to the symmetry of TCP-connections), with $\tilde{V}$ representing the nodes in the IPFS overlay and $\tilde{E}$ the connections among them. $\tilde{G}$ consists of *all* nodes, including clients.

Since clients do not contribute to the DHT, one of the core pillars of IPFS, we focus most of our analysis on the IPFS overlay without clients. Let $V \subseteq \tilde{V}$ be the set of IPFS nodes that are not clients and $G := \tilde{G}[V]$ the induced subgraph of V, i.e., the graph with vertex set V and all edges from $\tilde{E}$ that have both endpoints in V. V can be, to a large extent, learned by crawling the DHT buckets of each node, whereas it is not straightforward to reason about $\tilde{V}$ as client nodes are not registered anywhere or announced on the network.

What exactly can be learned by crawling each node's buckets? IPFS attempts to reflect every new connection (inbound or outbound) to DHT-enabled nodes (i. e., nodes that are not pure clients) in DHT buckets. When connections are terminated for whatever reason, the corresponding entry is deleted from the respective bucket in IPFS versions $<$ v0.5. Therefore, for v0.4. nodes, these node's buckets, and hence the DHT as a whole when we performed the measurement study (cf. Section 4.7), contains only active connections. On a side note, in newer versions of IPFS, routing entries are kept even if the connection was terminated. The routing table periodically (at least every 10 min) evicts peer entries that are not "useful", which currently means experiencing large round trip times (RTTs) or connection losses. One can argue that these changes should not significantly impact our considerations, as peers with terminated connections will be evicted from the routing table nevertheless with IPFS merely being more forgiving with respect to short connections outages.

If buckets were arbitrarily large, nearly all overlay links (E) would (also) be part of the buckets. However, buckets are limited to $k = 20$ entries. This leads to situations where connections are established but cannot be reflected in buckets. For example, during DHT lookups, IPFS establishes connections to all newly discovered nodes and attempts to add them to buckets. To avoid maintaining a nearly infinite number of connections, IPFS nodes start to randomly terminate connections that are older than 30 s once their total number of connections exceeds 900 (default value). If relevant buckets are full at both nodes, the connection exists without being reflected in any bucket at all. Let $G' = (V', E')$ be the bucket-induced subgraph of G, i. e., $E' = \{(v_i, v_j) \in E : v_j \text{ is in a bucket of } v_i\}$ and $V' \subset V$ is the set of all nodes used in E'.

For visualizing the possible configurations, Figure 24 depicts an example overlay network without clients (G), with the connections that are reflected in buckets (G') redrawn on top. A connection between two nodes can either

1. be reflected in buckets by both nodes (e. g., (v_2, v_3)),
2. be reflected in buckets by only one node (e. g., (v_0, v_1)) or
3. exist independently of bucket entries (e. g., (v_0, v_3)).

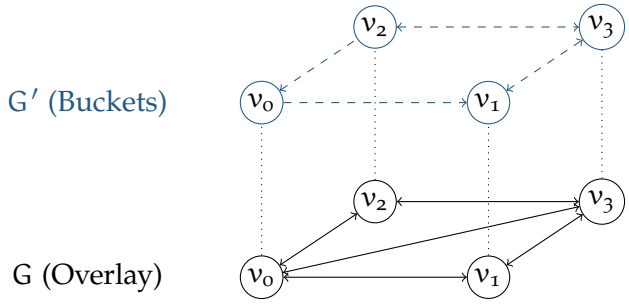


Figure 24: Types of connections: non-client overlay (bottom) and buckets (top).

In practice, it therefore usually holds that E' is a strict subset of E and G' therefore strict subgraph of G. Unlike G, G' is furthermore a *directed* graph. The direction is important since lookups of data or other nodes *only* consider the nodes stored in buckets.

In the case of a well functioning DHT it is expected that $V' \equiv V$ and $\tilde{V} \setminus V'$ is exactly the set of client nodes. Event though it is likely that not all node pairs from E are reflected in E', we expect that G' is strongly connected and that it is therefore, in principle, possible to enumerate all non-client nodes V by crawling the DHT. Our crawling results (cf. Section 4.7.1) support this assumption. Before delving into our crawling methodology and the remaining insights gained through it, we present an empirically founded estimation of the relationship between $\tilde{G}$, G and G'.

4.5 MEASURING THE INTERPLAY BETWEEN $\tilde{g}$, g AND g'

In the following, we focus on the default behavior of nodes (with and without NAT), the share of clients in the IPFS overlay (i.e., $\tilde{V} \setminus V$) and how much of the overlay of DHT-enabled nodes (G) is reflected in buckets (G').

4.5.1 *Empirical Results of Monitoring nodes*

First, we started an IPFS node, version v0.5.0-dev with default settings and a public IP address, without any firewall. Every 20 s, we queried the number of overlay connections, as well as the number of nodes in the node's buckets for a total of 3 d. For the number of overlay connections we relied on the API that IPFS offers in this case, whereas the number of nodes was extracted through our external crawling tool (discussed in Section 4.6). The results are depicted in Figure 25, which shows the number of overlay connections (edges from $\tilde{E}$, solid red line), connections with DHT-enabled nodes (edges from E, dashed green line) and the number of connections in the buckets (edges from E', dashed blue line). The measurement began simultaneously with

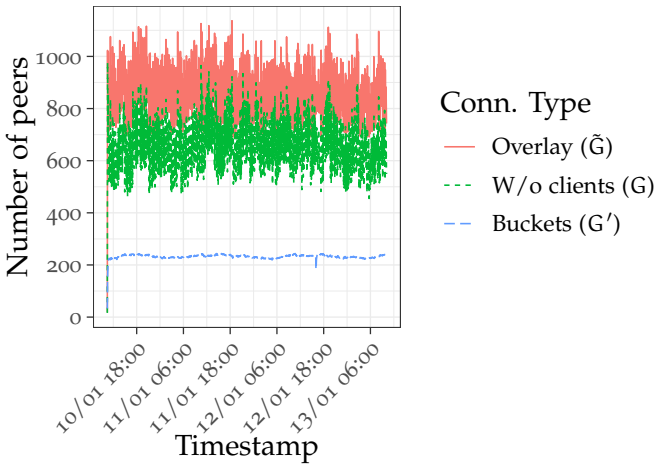


Figure 25: Number of connections of an IPFS node with default settings. We distinguish between all overlay connections, the overlay without clients and the buckets.

the IPFS node itself, hence, the short start-up phase in the beginning. It can be seen that the number of connections largely fluctuates around a value, with up to 400 connections established in just a few minutes. This behavior is due to the way IPFS handles its connection limit.

The default number of connections an IPFS node will establish is 900, but it does not cap at that value. Instead, IPFS 1) starts a new connection with every node it encounters when querying data and 2) accepts every incoming connection at first. Roughly, if the number of connections exceeds the limit, IPFS evicts connections (uniformly) at random that are older than 30 s, until the upper limit is satisfied again. For more information on the connection manager, the reader is referred to [220]. Furthermore, 76 % of connections are DHT-enabled, on average, indicating a notable difference between the overlay with clients ($\tilde{G}$) and the one without (G). We performed the same experiment for a node behind a firewall for a duration of 3 d, the results are shown in Figure 26. Several properties can be observed from both experiments. Firstly, a node behind a NAT has almost two orders of magnitude less connections than its non-NATed counterpart. Secondly, most connections of the non-NATed node are *inbound* connections, i.e., established from another peer on the network. Since our node was idle and not searching for content or other peers, it has no external trigger to start outbound connections. The NATed node cannot accept incoming connections, hence, the low number of connections. Note that the presented number of DHT-enabled connections is occasionally smaller than the number of connections corresponding to DHT bucket entries. The former is obtained via IPFS API functions that may lag behind in reporting the protocols of peers.

Last but not least, we are interested in a more holistic perspective on the different overlay types and number of nodes in the network.

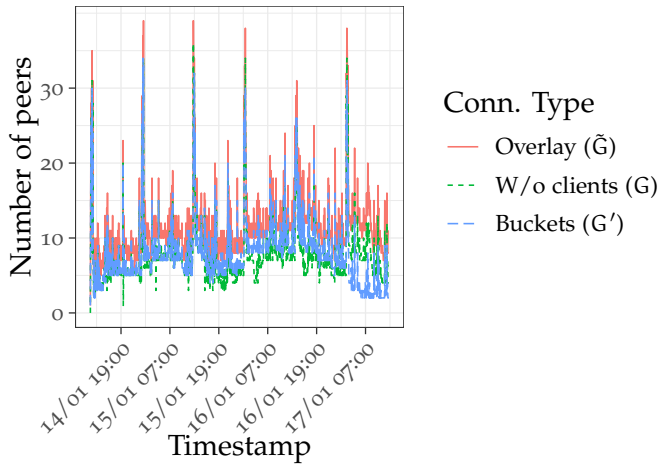


Figure 26: The same as Figure 25, but behind a firewall.

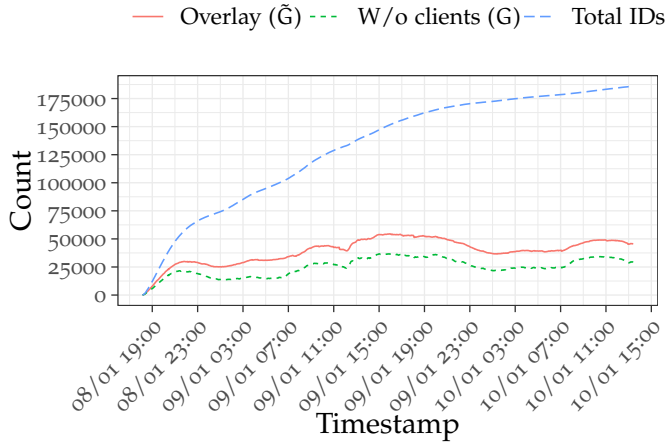


Figure 27: Number of IDs seen, overlay connections and connections to DHT-enabled peers of a node that has no connection limit.

To this end we started an IPFS node (v0.5.0-dev) with a public IP and a connection limit of 500000. Similar to the other experiments, we logged the connections every 20 s for a total of 2 d. The results are depicted in Figure 27, which shows the number of connections over time, the number of DHT connections and the total number of node IDs seen. On average, the node had $3.9 \cdot 10^4$ connections, 63.6 % of which were DHT connections. Again, the vast majority of these connections is inbound, since our node was idle. The number of node IDs is steadily increasing, whereas the number of connections is not, which could be due to a one-shot usage or people deleting the configuration directory, which holds the private key, to solve errors.

4.5.2 *Analysis of the Difference between* E *and* E'

In Figure 25 it is clearly visible that the buckets (E') store 22.2 % of all connections between DHT-enabled nodes (E), due to the buckets' limited capacity of $k = 20$ nodes (cf. Section 4.4). The connections that are not reflected in E' can, therefore, *not* be found by crawling the DHT nor be obtained through passive measurements alone. This raises the question: why is the gap between connections stored in a node's buckets (E') and all overlay connections between non-client nodes (E) so significant?

Analytically, we are interested in the following quantity: Given $N := |V|$ non-client nodes in the overlay, what is the expected number of nodes stored in the buckets? The distribution of nodes to buckets is highly skewed, since the first bucket is "responsible" for one half of the ID space, the second bucket for one fourth etc. [8, 168].

The expected number of nodes in each bucket i is therefore $\min\{20, N \cdot p_i\}$, with $p_i := 2^{-i}$. Although we do not encounter every peer equally likely, this reasoning still holds: During bootstrap, an IPFS node performs lookups for its own node ID as well as random targets tailored for each bucket. The former ensures that it knows *all* nodes in its direct neighborhood, partially filling the smallest, non-empty bucket. The latter ensures that it knows some nodes from each other bucket, filling them completely with high probability. Which bucket will be the smallest non-empty bucket therefore only depends on the total number of non-client nodes in the overlay.

Abusing notation, this yields:

$$\mathbb{E}[\#\text{ nodes in buckets}|N\text{ nodes in overlay}] \tag{31}$$

$$= \sum_{i=1}^{256} \mathbb{E}[\#\text{ nodes in bucket } i|N\text{ nodes in overlay}] \tag{32}$$

$$= \sum_{i=1}^{256} \min\{20, N \cdot p_i\}. \tag{33}$$

The result of this analytical consideration are depicted in Figure 28. The solid red line corresponds to a "perfect" setting, where each overlay connection would be stored in a bucket, whereas the dashed blue line is the result of Equation (33). Plugging the empirically found number of nodes from Section 4.7 into this formula yields an expected number of bucket entries between 232 and 247, which coincides with the measured average number of entries of 232.5 (max. 245).

So far, we've seen some indications on the relationship between the IPFS overlay $\tilde{G}$, the overlay without clients G and what is stored in buckets G'. In the following, we are interested in obtaining measurements of G' to learn more about the topology of the network.

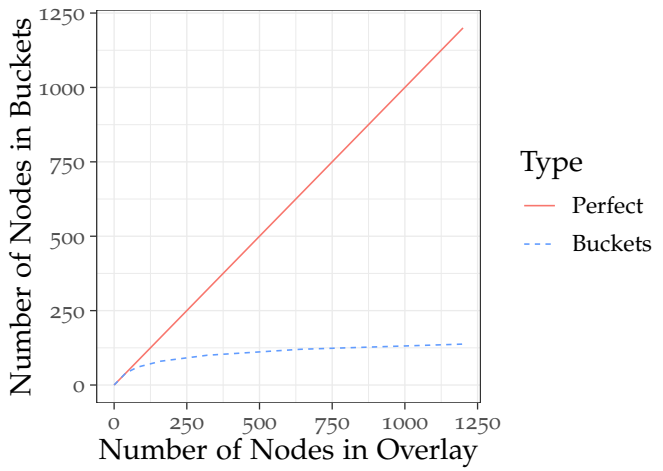


Figure 28: Analytical approximation: number of overlay connections vs. the number of nodes actually stored in the buckets.

4.6 CRAWLING THE KADEMLIA DHT

To crawl the Kademlia DHT, we specifically developed ipfs_crawler from scratch to generate snapshots of the IPFS overlay network. However, its potential applications transcend IPFS. IPFS' network layer is implemented in the libp2p networking library, which originated as part of the IPFS project but was modularized into its own standalone library. Therefore, ipfs_crawler can easily be adapted to crawl other libp2p-based networks.

4.6.1 *Why Yet Another Crawler?*

We resorted to developing our own crawler instead of re-using existing Kademlia crawlers [244] for two reasons: first, IPFS' protocol and handshake structure is highly complex and best used with the provided API, which would be hard to integrate in existing crawlers. Second, and more importantly, although the crawling *literature* on Kademlia is vast, there are virtually no open source implementations.

Therefore, we made our code public and are cooperating with IPFS/libp2p developers to incorporate the crawler into a periodic monitoring infrastructure, since ipfs_crawler is an effective tool to assess the state and health of the network. Additionally, the obtained data and insights from crawling may be useful for engineers and further research on, e.g., performance and resilience. As we previously outlined in Section 4.4, ipfs_crawler is able to enumerate all peers but cannot capture all links between those peers: in contrast to other Kademlia implementations, IPFS establishes a connection with every peer it encounters and maintains a large number of connections that do not correspond to any DHT routing table entries.

4.6.2 *Crawl Procedure*

As described in Section 4.3, nodes in the network are identified through a (multi-)hash of a public key. The crawler used new key pairs in every run to thwart potential biases due to re-occurring IDs. Moreover, an IPFS node will not store our crawling nodes in its buckets, since our nodes are marked as clients who do not actively participate in the Kademlia communication[64].

[64] *Nodes exchange a list of protocols they can serve. By not including the DHT protocol in said list, other nodes know that we cannot answer DHT messages.*

Nodes can be found by issuing `FindNode` packets for some target ID, i.e., the hash of a public key. To completely crawl a node, one has to send a `FindNode` packet for each possible bucket. This is due to the fact that a node returns its k closest neighbors to the target provided in a `FindNode` packet. The closest neighbors to the target are the ones in the bucket the target falls into. If the bucket in question is not full (i.e., less than k entries), the closest neighbors are the ones in the target bucket and the two buckets surrounding it (cf. Sections 2.2 and 3.6).

Since the bucket utilization of remote nodes is unknown, we do not know in advance how many requests per node we have to send for obtaining all of its DHT neighbors. ipfs_crawler, therefore, sends `FindNode` packets to targets with increasing common prefix lengths and stops once no new nodes are learned. This results in a significant speed improvement as no requests are sent for buckets that are nearly certainly empty (since the number of potential bucket entries decreases exponentially with the common prefix length). Faster crawls, in turn, enable us to capture more accurate snapshots of the dynamically changing network.

ipfs_crawler starts by connecting to the IPFS bootstrap nodes, collects their routing table content and successively tries to connect to every peer it has not tried before until no more new nodes were learned. Therefore, if the IPFS network were static and nodes replied to requests deterministically, a crawl would always yield the same results. Due to the inherent dynamics in the overlay this is not the case; instead, repeated crawls allow us to observe changes in the overlay over time.

As an example, consider Figure 29 which depicts the crawling process for one node and said nodes' bucket organization. The receiver's ID is 1101. The first bucket contains only peers whose ID xor 1101 starts with no leading zeros, hence, node IDs starting with 0. Similarly, for an ID to be stored in the second bucket, the xor has to start with one leading zero, yielding IDs starting with 10, and so on.

4.6.3 *Features*

For every peer it encounters, ipfs_crawler saves the following information:

- the peer ID, i.e., the (multi-)hash of a public key,

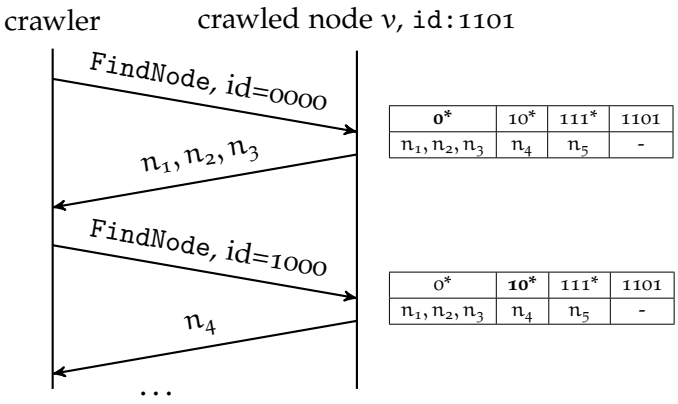


Figure 29: Sequence diagram of the crawl.

- all available addresses (e.g., IPv4, IPv6, relay, ...) of the peer and
- whether a connection could be established.

If a connection attempt was successful, it also includes

- the agent version (was not implemented for the Nov. 2019 crawl) and
- the content of the routing table entries as an edge list with timestamps when the respective edge was seen.

Taking the example in Figure 29, the edge list obtained from crawling node v would be $(v, n_1, t_1), (v, n_2, t_1), \ldots, (v, n_5, t_3)$.

ipfs_crawler enumerates the nodes in the network, but without ground truth it is hard to assess the quality and completeness of a crawl. Therefore, it might be desirable to perform a sanity check whether some pre-defined IPFS-nodes are found through crawling. These can be well-known nodes, such as the `ipfs.io`-gateway, or self-run nodes, specifically started for the purpose of sanity checking the results. If provided, ipfs_crawler checks if it found the respective nodes and notifies the user in case it was not successful.

Furthermore, if configured, ipfs_crawler will cache the nodes it has seen. The next crawl will then not only start at the bootstrap nodes but also add all previously reachable nodes to the crawl queue. This caching additionally increases the crawl speed, since it overcomes the bottleneck of finding peers to connect to in the beginning of each crawl.

4.6.4 *Hash Pre-Image Computation*

Unlike more canonical Kademlia implementations, whenever an IPFS node receives a `FindNode` packet, it hashes the provided target and searches for the nearest neighbors to that hash[65]. Therefore, to know which target to send for which bucket, we need to compute pre-images

[65] *Similar to Ethereum, cf. Section 3.3)*

that, when hashed, yield the desired common prefix length between the `FindNode` target and the node ID of the node we are crawling. To that end, we generated pre-images for every possible prefix of 24-bit length. In other words, we computed 2^{24} pre-images such that, for each possible prefix of the form $\{0,1\}^{24}$, there is a hash starting with that prefix.

Equipped with this table, one lookup operation is sufficient to pick a pre-image that, when hashed by the receiver, will yield the desired bucket. Note that this table can be used for an arbitrary number of crawls, hence, the computation only had to be performed *once*.

4.7 CRAWLING RESULTS

We repeatedly crawled the IPFS network from 2019-11-26 until 2019-12-03 for a total duration of 7 d. We denote this set of crawls by C_1 which will be the main topic of this section. Additionally, we conducted crawls for a two week period in the beginning of 2021 from 2021-01-24 until 2021-02-07, which we will denote by C_2. If reasonable, the results from C_1 will be complemented by ones from C_2, e.g., to illustrate changes in the network. The complete report on C_2 can be found in the Appendix (cf. Section 7.1.1). Crawls were started one at a time and back to back, in the sense that as soon as a crawl was finished, a new one was started. We performed a total of 2400 crawls in C_1, with a single crawl taking an average of 4 min to complete. In C_2, 5039 crawls were performed with a similar average duration per crawl.

The most important aspect of every crawler is its speed, as it directly influences the accuracy of obtained snapshots [248]. Since the overlay is changing during crawls due to peer churn, the longer a crawl takes, the higher the risk of unwanted artifacts in the snapshots [249]. Therefore, our ipfs_crawler is optimized for small crawl times and is able to crawl 50000 nodes in roughly 4 min, on average. Figure 30 depicts a boxplot summarizing the distribution of crawling times for the two crawl sets in November 2019 and February 2021. The box corresponds to the 25 % and 75 % quartiles, respectively, with the median shown as the solid line. It can be seen that we were able to significantly reduce the variance of the crawler's runtime, although sacrificing some performance in the process. Nevertheless, despite the high variance in C_1, 75 % of crawls took less than 5 min to complete and only 2.7 % of crawls took longer than 10 min. Hence, most crawls from C_1 were quick, therefore adhering to the principles of Stutzbach and Rejaie [248], allowing for confidence in the validity of the obtained snapshots.

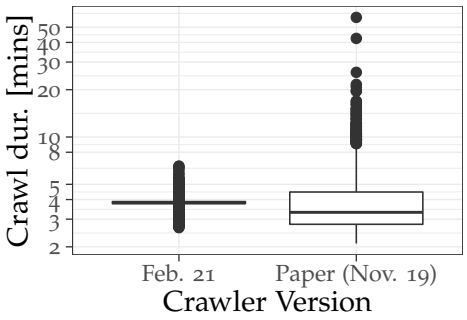


Figure 30: Boxplot of crawl durations of crawls from C_1 in Nov. 2019 and from C_2 in Feb. 2021.

Session Duration	Percentage	Number of sessions
5 minutes	56.92	2.188e6
10 minutes	26.11	1.00e6
30 minutes	2.68	1.0e5
1 hour	0.43	1.6e4
1 day	< 0.01	32
6 days	< 0.01	2

Table 2: Inverse cumulative session lengths: each row gives the number of sessions (and total percentage) that were *longer* than the given duration.

4.7.1 *Number of Nodes, Reachability and Churn*

To get an idea for the size of the network, we first focus on the number of nodes in the network and their session lengths. During our 7 d crawl, we found a total of $3.09 \cdot 10^5$ distinct node IDs, with an average number of $4.4 \cdot 10^4$ per crawl. This is consistent with the results obtained in Section 4.5, hinting that both methods provide an adequate view of V, the set of non-client nodes in the IPFS overlay. Surprisingly, of all the nodes that were queried, the crawler was only able to connect to 6.6 %, on average.

We suspect that most IPFS nodes in C_1 are run by private people connected through NAT. This hypothesis is supported by our results: about 52 % of all nodes report only local IP addresses for other nodes to connect to, which is exactly the behavior of nodes behind symmetric NATs (cf. Section 4.7.2). Note that in IPFS versions $\geqslant$ v0.5 reporting local IPs to the DHT is uncommon. Instead, nodes start as clients and only actively participate in the DHT if they succeed at finding their externally reachable IP address and port. Regarding C_1, if most nodes are behind NATs, i.e., clients, they are also prone to short uptimes, since these are probably used as client nodes which are shut down after use.

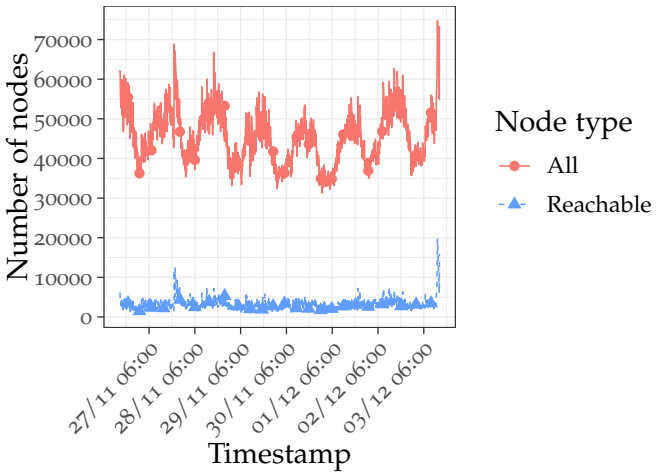


Figure 31: Number of nodes over time, distinguished by all and reachable (=answered to our query) nodes. Times are in UTC.

Exactly this behavior can be observed regarding the session lengths, which are depicted in Table 2. We define a session as the time difference between the crawl, when we were able to reach the node and when it became unreachable again. The table depicts the inverse cumulative session lengths: each row holds the number of sessions (and their percentage) that were longer than the given duration. For example, roughly 56 % of all sessions were longer than 5 min, or equivalently, 44 % of all sessions were *shorter* than 5 min. Hence, participation in IPFS in C_1 was dynamic and rather short-lived. In comparison, computing the session lengths of C_2 in the same fashion shows (cf. Table 11) much longer sessions, i.e., 40 % were longer than 30 min, 25 % even longer than one hour. This yields an interesting insight, as the significant increase in session lengths correlates with the exclusion of clients behind NATs from the DHT; indicating that the remaining peers might operate as always-on infrastructure. Note that these infrastructural nodes serve different purposes: while ProtocolLabs and other entities operate DHT boosters[66], a closer inspection of the agent versions indicates that between 25 % and up to 60 % of DHT-enabled nodes are part of the storm botnet [224] (cf. Section 7.1.1). Note that in Nov. 2019 and even the beginning of 2020, storm nodes did not play a role, the emergence of these nodes commenced in mid-2020. For a side note on storm nodes, the reader is referred to the Appendix (cf. Section 7.1.1.1).

[66] *https://github.com/libp2p/hydra-booster.*

We also observed a periodic pattern in the number of nodes found through crawling, as shown in Figure 31. The figure distinguishes between all nodes and nodes that were reachable, i. e., the crawler was able to establish a connection to these nodes. The significant increase in the number of nodes at the end of the measurement period could stem from other researchers' nodes or even an attack on the network. It can be seen that between noon and 7pm UTC, the number

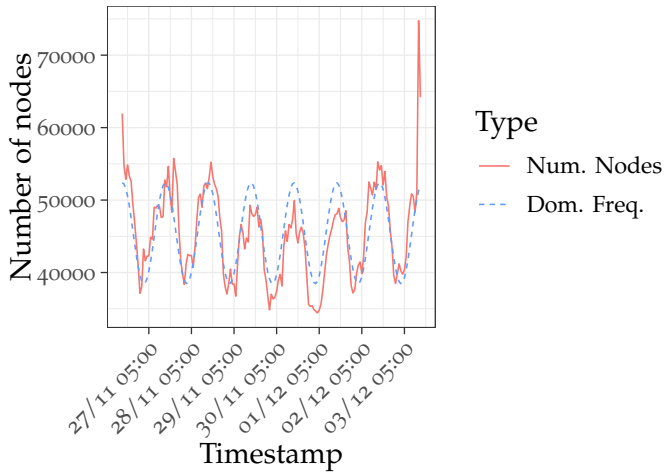


Figure 32: Hourly resampled Figure 31 with dominant Fourier-frequency.

of nodes increases significantly. Interestingly, the periodic pattern has indeed a period of 24 h, as a Fourier analysis reveals[67], illustrated in Figure 32. The figure depicts the same data as in Figure 31 but re-sampled to hours, as well as the dominant Fourier frequency, which has a frequency of 24 h. This might hint at a usage pattern in that users start their IPFS nodes on-demand in their spare time and shut them down after use. Additionally, this underlines the hypothesis of most nodes being operated by private people behind NATs, as the number of *reachable* nodes, i. e., nodes we could connect to, does not fluctuate as much. Although IPFS' handling of nodes behind NATs has significantly improved since C_1in Nov. 2019, similar patterns in the number of nodes over time can also be observed for C_2 (cf. Figure 46). However, the the fluctuations are less prominent and show a mixture of a trend as well as other frequencies, with the dominant Fourier frequency still being 24 h. One could argue that private users behind NATs will tend to use their nodes in their spare time in the evening. Hence, a usage pattern with peaks in the afternoon of UTC time hints to many users in Asia, since UTC afternoon corresponds to evening times in, e. g., China. A superposition of frequencies as observable in C_2, could stem from peaks in IPFS-usage from different countries; a hypothesis that can be justified by the distribution of nodes over countries.

[67] *A big "thank you" to Jochen Fink for his support with the Fourier analysis.*

4.7.2 *Node Distribution over Countries and Protocol Usage*

Table 3 depicts the top ten countries, both for all discovered nodes and for nodes that were reachable by our crawler[68] These ten countries contain 90 % (93.7 % in the case of reachable nodes) of the whole

[68] *We use the GeoLite2 data from MaxMind for this task, available at* `https://www.maxmind.com`.

All			Reachable		
Country	Count	Conf. Int.	Country	Count	Conf. Int.
LocalIP	23900	±173.9	US	1720	±26.7
CN	5630	±11.3	FR	630	±17.7
DE	4220	±33.4	DE	569	±8.8
US	4100	±54.1	CA	119	±2.3
FR	1390	±33.3	PL	73	±2.1
CA	970	±19.9	CN	58.6	±0.5
PL	690	±18.0	GB	26.6	±0.2
IL	320	±2.6	SG	20.9	±0.2
GB	171	±0.9	IL	15.0	±0.4
HK	168	±2.0	NL	12.9	±0.1

Table 3: The top ten countries per crawl, differentiated by all discovered nodes and nodes that were reachable. Depicted is the average count per country per crawl as well as confidence intervals.

network, already hinting at centralization tendencies regarding the spatial distribution of nodes.

Again, it can be seen that 52 % of all nodes *only* provide local or private IP addresses, thus making it impossible to connect to them. This is in line with the default behavior of IPFS versions $<$ v0.5, when operated behind a NAT. When a node first comes online, it does not know its external IP address and therefore advertises the internal IP addresses to peers it connects to. These entries enter the DHT, since IPFS aims to provide support for building private IPFS networks. Over time, a node learns about its external multi-addresses (multi-addresses contain address and port, cf. Section 4.3) from its peers. An IPFS considers these observed multi-addresses reachable, if at least four peers have reported the same multi-address in the last 40 minutes and the multi-address has been seen in the last ten minutes. This is never the case for symmetric NATs, which assign a unique external address and port for every connection, yielding a different multi-address for every connected peer. As mentioned before, if node cannot successfully obtain an externally reachable and stable IP address/port pair, they will refrain from acting as DHT enabled nodes and therefore not advertise their local IP addresses to the network.

Furthermore, there is a significant difference visible especially between China and the U.S.: although most IPFS nodes are in China, the vast majority of reachable nodes stems from the U.S. The notable difference could stem from the great firewall of China, which severely restricts the Internet access of Chinese users.

In comparison, in C_2 from Feb. 2021, nodes with only local IPs do not play a role anymore. Interestingly, more nodes from Korea and Hong Kong emerged — possibly related to the political conflict and on-going protests between Hong-Kong and China.

Protocol	Perc. of peers	Abs. count
ip4	> 99.9	3.09e5
ip6	80.5	2.49e5
p2p-circuit	1.5	4.7e3
ipfs	1.3	3.9e3
dns4	< 0.1	207
dns6	< 0.1	93
dnsaddr	< 0.1	12
onion3	< 0.1	1

Table 4: Protocol Usage.

As stated in Section 4.3, IPFS supports connections through multiple network layer protocols; Table 4 shows the prevalence of encountered protocols during our crawls. If a node was reachable through multiple, say IPv4 addresses, we only count it as one occurrence of IPv4 to not distort the count. The majority of nodes support connections through IPv4, followed by IPv6, whereas other protocols are barely used at all. The protocols "ipfs" and "p2p-circuit" are connections through IPFS' relay nodes, "dnsaddr", "dns4/6" are DNS-resolvable addresses and "onion3" signals TOR capabilities. Similar results can be observed for C_2, with a decrease in the number of IPv6-capable addresses.

4.7.3 *Graph Properties & Overlay Topology*

A crawl yields a view on the nodes V' in the buckets of non-client nodes, and their connections to each other, i.e., E'. As discussed in Section 4.4, we obtain a measurement of V' (which is approximately V) and E' which is a strict subset of E. Nevertheless, $G' = (V', E')$ is the graph used by IPFS to locate data and nodes through the DHT and therefore provides a lot of insights. Since there is a huge discrepancy between all discovered and the reachable nodes, and since we cannot measure the properties of unreachable nodes, we distinguish these two classes in the analysis.

4.7.3.1 *Degree Distribution*

Commencing with studying the degrees of nodes, Figures 33 and 34 depict the cumulative log-log in-degree distribution for the crawl on the 1st of December 2019, 00:00 UTC; note that other crawls yield similar results. Aggregated degree statistics over all crawls are summarized in Table 5. We differentiate between all found nodes (Figure 33) and only reachable nodes (Figure 34). The (roughly) straight line in the figure corresponds to a highly skewed distribution where some peers have very high degree whereas most peers have a small degree. The shape of the distribution indicates the possibility for a heavy-tailed power-law distribution of degrees, as observed in other Kademlia

studies [232]. To test the power-law hypothesis, we proceed as outlined by Clauset et al. [59]: (1) estimate the power-law parameters x_{min} and α using a maximum-likelihood approach, (2) calculate the goodness-of-fit through bootstrapping[69] and (3) compare the power-law hypothesis to other alternative distributions using Vuong's test [119]. For the number of bootstrapping runs, we adhere to [59], i.e., for p-values with 2 decimal digit accuracy, 2500 simulation runs are advised[70].

[69] *Bootstrapping involves re-sampling the sample data with replacement to quantify uncertainties of estimates [88].*

[70] *"[A] good rule of thumb turns out to be the following: if we wish our p-values to be accurate to within about ϵ of the true value, then we should generate at least $\frac{1}{4}\epsilon^{-2}$ synthetic data sets [=bootstrapping runs]."*

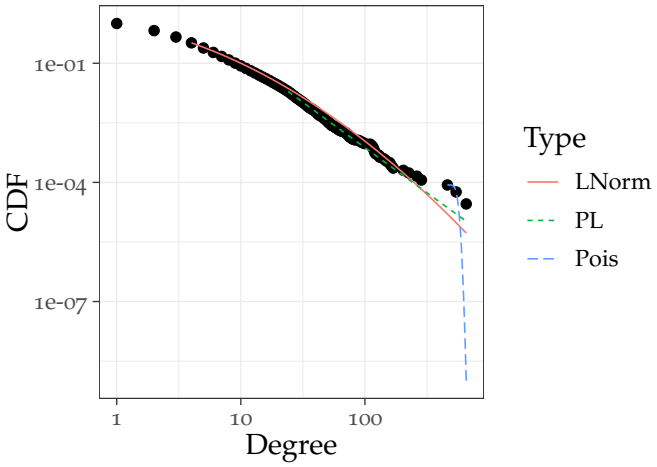


Figure 33: In-Degree distribution from the crawl on 1st of December 2019, 00:00 UTC, including *all* found nodes. Other crawls yield similar shapes.

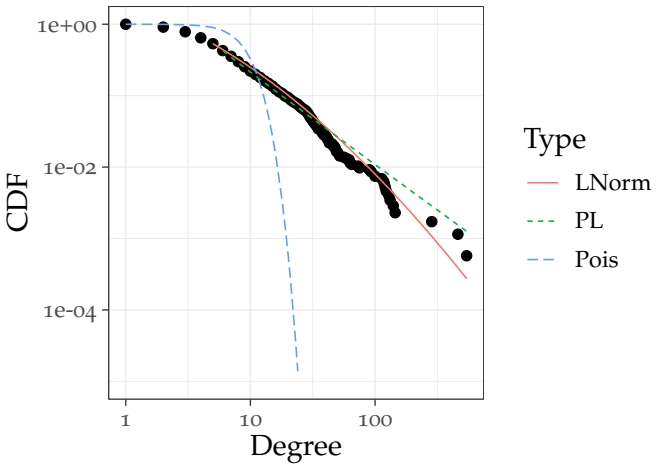


Figure 34: The same distribution as Figure 33, but only including *reachable* nodes.

In Figures 33 and 34 three candidate distributions were fitted to the data: power-law, log-normal and Poisson. By visual inspection alone, both power-law and log-normal distributions seem viable as potential distribution candidates. For Figure 33, both power-law and

log-normal yield a p-value of $p > 0.1$ for step (2), the goodness-of-fit ($p_{PL} = 0.75, p_{LN} = 0.17$). Therefore, in accordance with [59], both distributions, power-law and log-normal, are potential candidates. A comparison of both distributions using a log-likelihood ratio test yields a p-value of $p = 0.46$, meaning that we cannot reject the null-hypothesis that both distributions are equally far from the true distributions of the data [119]. In other words, the observed degree distribution may have plausibly be sampled from a power-law or a log-normal distribution.

For the degree distribution of reachable nodes in Figure 34, the situation is less ambiguous. Again, following the three steps of testing a power-law hypothesis by Clauset et al. [59], we obtain a significant p-value for the log-normal distribution but not the power-law one ($p_{PL} = 0.04 < 0.1, p_{LN} = 0.33 > 0.1$). Similarly, when comparing the two hypotheses in step (3), the computation yields a preference for the log-normal distribution, with a p-value of $p = 0.07 < 0.1$ and a likelihood ratio in favor of the log-normal distribution — allowing us to reject the power-law hypothesis.

These results are in contrast to prior measurements on Kademlia systems which indicate a power-law distribution of degrees [232]. The difference could be due to the limitations of the crawling methodology itself (cf. Section 4.4), or stem from actual differences of IPFS in comparison to other Kademlia systems. Although the degree distributions are not exact power-laws and the network can, by this measure alone, not be considered scale-free, IPFS' overlay (for C_1) still exhibits similar properties as scale-free networks, as we will see in Section 4.7.3.2.

	Min.	Mean	Median	Max.
total-degree	1	14.3	10.2	942.2
in-degree	1	7.2	3.0	935.5
out-degree	0	7.2	6.5	53.6

Table 5: Average of the degree statistics of all 2400 crawls.

Regarding the set of crawls from Feb. 2021, i.e., C_2, the degree distributions have dramatically different shapes (cf. Figure 48 in the Appendix). Nodes in C_2 have a significantly higher degree on average, leading to a much more shallow CDF for small degrees. Interestingly, about one third of all nodes lie within the degree interval $[200, 230]$, leading to significant differences between C_1 and C_2, as we will see in the resilience analysis (cf. Section 4.7.3.3). This stems, among potential other reasons, from two factors: (1) significant improvements of the libp2p networking library and our crawler and (2) longer session lengths. Regarding (1), since IPFS v0.5, the Kademlia buckets are populated much more rigorously and quickly after startup. Furthermore, refreshments at least every 10 min lead to more nodes in buckets than for older IPFS versions. We also improved our crawler, especially the interplay with libp2p, leading to a higher throughput of crawl requests

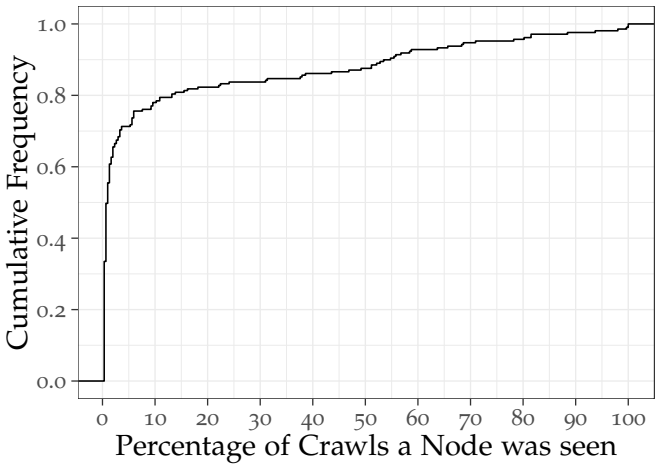


Figure 35: ECDF of how often the same nodes were within the top-degree nodes.

and more reliability, which positively influences the number of edges the crawler finds. In summary, in comparison to C_1, the crawler is able to enumerate V' evenly well as for C_1, but performs significantly better on enumerating E'. Simultaneously, session lengths have become significantly longer (clearly visible in Table 11 in the Appendix), potentially due to more infrastructural nodes. Longer session lengths correspond to longer uptimes and therefore more populated Kademlia buckets.

Moving on from the general degree distribution, we focus our analysis on the top degree nodes in the following. Top degree nodes were defined as the 0.05 % of all nodes, yielding 22.2 nodes on average for C_1. We refrain from defining a number, say the 20 nodes with the highest degree, since this would weigh crawls with fewer nodes differently than crawls with a higher number of total observed nodes. Figure 35 depicts a cumulative distribution of how many times the nodes with the highest degree were seen in the crawls. If a node was in the set of highest-degree nodes in one run but not in other runs, its "percentage seen" would be $\frac{1}{\text{\# of crawls}} = \frac{1}{2400} = 0.0033$ or 0.33 %. On the other extreme, if a node was within the highest degree nodes in every crawl, its percentage seen would be a 100 %.

The cumulative distribution in Figure 35 shows a high churn within the highest degree nodes: approximately 80 % of nodes were only present in 10 % of the crawls. Only a few nodes have a high degree in the majority of all crawls; these nodes are the bootstrap nodes along a handful of others. C_2 differs in that 40 % of top degree nodes were found in the majority of crawls, further underlining the hypothesis of more infrastructural nodes in the DHT. Again, the reader is referred to the Appendix in Section 7.1.1 for the complete crawl report on C_2.

4.7.3.2 *Graph Metrics*

The degree distribution already gives us insights into the properties and topology of the overlay graph, i.e., that its degree distribution is highly skewed and is at least similar to scale-free networks. In the following we deepen our understanding of IPFS' overlay topology through comparisons of various graph metrics to Erdős-Rényi (ER) [93] and Barabási-Albert (BA) random graphs [27] of the same size. ER graphs are a family of random graphs with a fixed number of nodes N and edges M, where the M edges are distributed among the potential "edge slots" equally likely. Asymptotically, ER graphs are equivalent to Gilbert random graphs [118], which also start with a fixed number of nodes but populate every edge with a probability p, inducing variability in the number of edges even for fixed p [93]. The degree distribution of ER random graphs follows a Poisson distribution, which differs from real-world networks such as the Internet, the world wide web and the IPFS overlay.

Therefore, Barabási-Albert proposed an alternative method[71] for creating random graphs, namely through preferential attachment [27], i.e., newly joined nodes are more likely to establish a link to nodes with a high degree. Graphs generated through preferential attachment display similar properties as scale-free networks, since, due to the attachment process, the degree distribution is asymptotically heavy-tailed [42].

[71] *Preferential attachment models and the occurrence of power-laws in real-world data has been studied before Barabási-Albert [42].*

Note that there is one caveat when studying the IPFS overlay: should the graph be studied as a directed or an undirected graph? Nodes establish TCP/QUIC connections, therefore, an undirected graph may seem reasonable. However, due to Kademlia and the resulting property that one node can have a routing table entry to another node but not necessarily vice versa (cf. Section 4.4), a directed graph may seem more appropriate (and has been used in the past for other Kademlia-based systems [232]). Unfortunately, overlay connections also transport BitSwap messages, the second method of obtaining content in IPFS, which are broadcast to all neighboring peers — hence suggesting an interpretation as an undirected graph. We resolve this conflict by adopting a hybrid perspective: in the following, we will treat IPFS as a *directed* graph in the comparison to other graph families, as metrics as the, e.g., average path length and betweenness centrality are mostly affecting Kademlia lookups. In the resilience analysis, i.e., the resistance of the overlay against random and targeted failures, we treat the graph as undirected, as in that case BitSwap plays a role as well.

For the comparison, we considered the following metrics: the average path length L, the diameter D, the global clustering coefficient C and the maximum betweenness centrality $\max(b_i)$. The average path length is defined as the mean of the length of shortest paths between all pairs of nodes [42]. Let $G = (V, E)$ be a graph with $|V| = N, |E| = M$

and let d_{ij} denote the shortest path from node i to j. Then the average path length is commonly defined as:

$$L = \frac{1}{N(N-1)} \sum_{i,j \in V, i \neq j} d_{ij}. \tag{34}$$

The diameter D is defined as the longest shortest path. For the clustering coefficient C, also referred to as transitivity[72], we follow the definition of [42]. Intuitively, the clustering coefficient C gives insights in how interconnected the neighborhood of a node v is, i.e., the interconnectedness between neighbors of v. More formally:

[72] *Note that for the transitivity, the direction of a graph is irrelevant.*

$$C = \frac{3 \cdot \text{number of triangles}}{\text{number of connected triples of vertices}}. \tag{35}$$

The betweenness centrality of a node v measures the importance and centralization of a node, by considering the number of shortest paths that traverse through v in ratio to all possible shortest paths between any two nodes in the network [42]. The maximum betweenness is therefore an indicator of the dominance of the most centralized node (according to this metric). The betweenness centrality is defined as follows, where n_{jk} is the number of shortest paths between j and k and $n_{jk}(i)$ the number of shortest paths that traverse i:

$$b_i = \sum_{j,k \in V, j \neq k} \frac{n_{jk}(i)}{n_{jk}}. \tag{36}$$

The results of the comparison are gathered in Table 6. For the analysis we chose the same measured graph as in the previous analyses, i.e., from the 1st of December 2019 at 00:00 UTC. The "GType" column depicts the corresponding graph, Erdős-Rényi (ER), Barabási-Albert (BA) or IPFS, "Metric" refers to the previously introduced metrics and "All"/"Reach." refers to the two different types of IPFS graphs. The latter two columns show the rounded mean and the respective confidence intervals when appropriate. For the corresponding ER and BA random graphs we generated ten graphs, respectively, and computed the mean of each metric as well as 95 % confidence intervals.

It can be seen that in terms of average path length IPFS resembles the BA graphs more closely than ER graphs, which is expected given the resemblance of IPFS's (C_1) degree distribution to a power-law. As we will see in Section 4.7.3.3, IPFS responds to random failures and targeted attacks similarly as scale-free networks, further underlining the similarities. Simultaneously, the clustering coefficient of the measured overlay is significantly higher than for both random graph models. This is in line with other results on diverse networks found in the real world, e.g., power grids, neural networks, the Lightning network and even actor collaborations [42, 132, 228, 266]. Common to all these networks is the "small-world" property, i.e., a high clustering

GType	Metric	All	Reach
ba	nodes	34856	1744
ba	edges	139414	15651
ba	apl	1.05±0.06	1.03±0.03
ba	trans	3e-04	0.014
ba	betw	< 1e-04	1e-04
ba	diameter	3±0.67	2.6±0.6
er	nodes	34856	1744
er	edges	148100	16157
er	apl	7.37	3.60
er	trans	3e-04	0.01
er	betw	0.001	0.005
er	diameter	15.5±0.38	6.3±0.35
ipfs	nodes	34856	1744
ipfs	edges	148100	16157
ipfs	apl	4.25	4.08
ipfs	trans	0.01	0.03
ipfs	diameter	10	9
ipfs	betw	0.003	0.062

Table 6: Comparison of measured graphs of the IPFS overlay at 1.12.19, 00:00 UTC with Erdős-Rényi & Barabási-Albert random graphs.

of edges which leads to a logarithmic growth of the diameter of a graph in the number of nodes N [266]. Note that a low diameter also implies a low average path length, but not vice versa. Humphries and Gurney [132] propose to compare the measured network g to the ER model r in terms of clustering coefficients (C_g, C_r) and average path lengths (L_g, L_r). These ratios are used to quantify "small-worldness" as

$$S := \frac{\gamma}{\lambda}, \text{ with } \gamma = \frac{C_g}{C_r}, \lambda = \frac{L_g}{L_r}. \tag{37}$$

When $S \gg 1$, the studied graph is deemed small-world. For IPFS, this yields $S_{all} = 74 \gg 1$ and $S_{reach} = 3 > 1$, allowing a classification as small-world, though S is moderately small in comparison to, e.g., actor collaboration networks [132].

Comparing to the results of C_2 in the Appendix (cf. Section 7.1.1.3), we can see a smaller number of nodes but about one order of magnitude more edges. As outlined above, this is, among other factors, due to improvements in the libp2p networking library and our crawler as well as longer session lengths. In comparison to C_1, the graphs from Feb. 2021 exhibit, unsurprisingly, a higher clustering coefficient, smaller average path lengths and diameters. This significant difference between graphs from C_1 and C_2 has an impact on the tolerance to

random failures and targeted node removals as we will see in the following.

In summary, IPFS' overlay in C_1 resembles scale-free networks in that the degree distribution is highly skewed with few peers having a high degree and many peers with a low degree. Other graph metrics like the diameter and average path length underline this similarity. Furthermore, IPFS shows a higher clustering of edges than random graphs and can therefore be considered a small-world graph. In the following, we will see that IPFS graphs from C_1 behave similarly to random failures and targeted attacks as scale-free graphs, whereas graphs from C_2 are remarkably tolerant to targeted attacks.

4.7.3.3 *Resilience Analysis*

The resilience of empirically-obtained graphs against random failures and targeted attacks has been subject of extensive investigation [12, 42, 232, 250]. To this end, the influence of deleted vertices or edges on graph properties like diameter, average path length and size of the largest connected component is studied. This gives insights into how well the respective graph tolerates failures and attacks which can in turn be leveraged to draw conclusions about, e.g., the availability of services. For example, Albert et al. [12] study, among others, the graph of the Internet at router and inter-domain level. While failures are tolerated well, i.e., the diameter barely changes, which is used as an indicator that performance will not degrade noticeably, targeted attacks dramatically increase the diameter[73].

[73] *Actually, the Internet is a special case, as it adapts to outages automatically, yielding a case of dynamic resilience [42].*

Failures are modeled through randomly (usually uniformly distributed) deleting vertices or edges, the former being much more widespread, from the graph under investigation. Targeted attacks correspond to deleting the "most valuable" vertices, which are normally the highest degree nodes although definitions may very depending on the context [228].

In [12], it is concluded that the Internet (or the world wide web) is tolerant to random failures due to the scale-free property, i.e., the skewed degree distribution. Randomly selecting nodes will, with high probability, yield a low-degree node whose removal does not significantly affect other nodes (and paths) in the graph. Furthermore, most paths traverse through one or multiple hubs with high degrees, increasing the robustness against random failures. These infrastructural nodes, however, render these graphs highly vulnerable to targeted attacks through systematic removal of the highest degree nodes. Through removing these nodes, paths originally running through them are prolongated and, depending on the interconnectedness, even partitions may occur. In contrast, ER random graphs are much more vulnerable to failures, as the degree distribution is not as skewed and each node contributes more evenly to the graph, i.e., ER random graphs lack large hubs. On the other hand, the lack of hubs

makes them robust against targeted attacks. Peer-to-peer networks are in many cases similar to scale-free graphs in terms of graph metrics but also resilience [228, 232, 250], with the exception of Gnutella which tolerates targeted attacks surprisingly well [250].

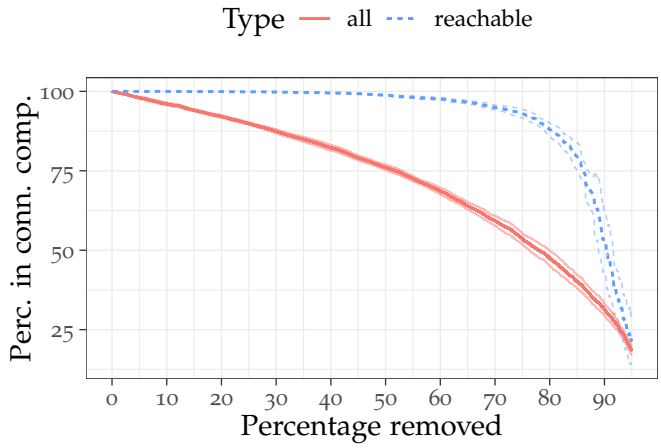


Figure 36: Resilience of the measured graph (same as above) to random removals, distinguished by all and reachable nodes.

For our analysis of IPFS, we simulate failures through randomly deleting vertices, and attacks by deleting the nodes with the highest degree. As a measure of the graph's health, we consider the fraction of nodes in the largest connected component, i.e., the fraction of nodes that remain connected after the removal of vertices. For the analysis we now treat the graph as *undirected*, as BitSwap requests between nodes can be transmitted in both directions due to the reliable overlay connection (cf. Section 4.7.3.2).

The results for random failures are depicted in Figure 36, distinguished by all nodes and the graph containing only reachable nodes. On the x-axis we see the fraction of removed nodes, i.e., ranging from 0 % to 95 % of all nodes in the original graph. On the y-axis is the fraction of remaining nodes in the largest component, e.g., if the graph has 100 vertices after the deletion of some, and 80 of those remaining nodes reside in the largest connected component, then the respective fraction is 80 %. We performed ten runs, the mean of which is depicted by the thicker line in the middle, whereas the opaque lines at the sides correspond to 95 % confidence intervals. The differences between the graph of all nodes and the one restricted to only reachable nodes could plausibly stem from the higher degree of connectedness in the latter graph. Another factor might be the more homogeneous degree distribution of the reachable graph, thus, removing a node is not as severe [232]. As seen in Section 4.7.3.2, the graph of reachable nodes has a higher clustering coefficient and maximum betweenness centrality, i.e., more paths go through central nodes, making it more robust against random failures. In general, IPFS' resilience against

outages is good, as removing 80 % of all nodes still leaves a connected component containing 50 % for the graph of all nodes and 75 % for the graph of reachable nodes, respectively. These results are comparable to similar considerations on the KAD [232] and Gnutella [250] overlays.

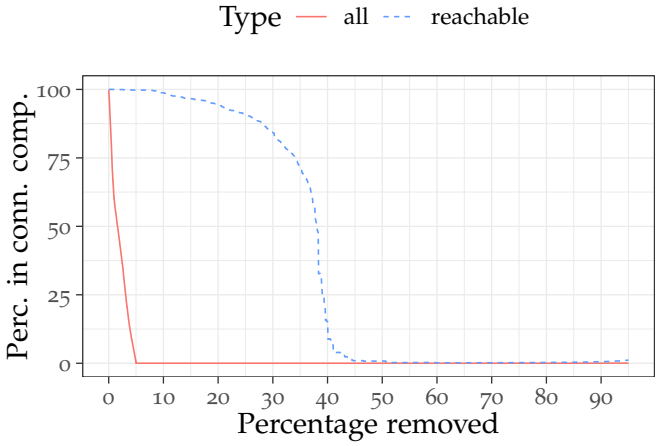


Figure 37: Resilience of the measured graph (same as above) to targeted removals, distinguished by all and reachable nodes.

In Figure 37, the same information as in Figure 36 is depicted for targeted removal of nodes with the highest degree. It can be seen that (1) both types of graph (reachable vs. all nodes) differ significantly and (2) attacks are tolerated less than random outages. The graph of all nodes has many satellite nodes, i.e., vertices that are connected to a node with high degree but were themselves not crawlable and therefore have no other edges. Removing high degree nodes thus quickly leads to a larger number of components of size 1 which is reflected in the low fraction of nodes that remain in one large connected component. In contrast, the graph of reachable nodes has, as we have seen above, a more homogeneous degree distribution — removing high degree edges does therefore not lead to an immediate partitioning as in the graph of all nodes. Still, the degree distribution is sufficiently skewed such that removing 30 % to 40 % of the high degree vertices quickly breaks down the largest connected component.

Interestingly, the resilience analysis for the IPFS overlay from Feb. 2021 yields dramatically different results (cf. for a discussion of potential explanations Section 7.1.1) in that the graph of reachable nodes tolerates failures and targeted attacks equally well, at least with respect to maintaining most nodes in one connected component. In that, the results are more comparable to the results on the Gnutella overlay [250] than the, in principle, more similar KAD network [232].

4.8 INTERIM CONCLUSION

IPFS is a hybrid between structured and unstructured overlays; sacrificing performance for improving robustness by combining a DHT with broadcasting of data requests for content retrieval. In the first half of this chapter, we've presented several approaches towards mapping IPFS overlay connections based on extracting information from the DHT. To this end, we employed monitoring nodes and a crawler specifically designed for taking snapshots of the IPFS overlay network. While not every edge of the overlay graph is observable through crawling, we argue that the obtained topologies are still valuable as they present a lower bound on the interconnectedness of IPFS nodes.

Through our monitoring nodes and a 7 d crawl (from 2019-11-26 until 2019-12-03) of the IPFS DHT, we obtain a holistic view on IPFS' node population as well as indicators about the larger overlay structure in terms of topology, geographical and protocol distribution. We furthermore performed a two-week crawl in February of 2021 and compared changes of IPFS in the recent past to our measurements. In particular, we measured[74] how many and what type of nodes participate in the network. While infrastructural nodes exist, short session lengths as well as daily fluctuations in the number of nodes indicate, among other factors, that a large share of nodes is operated by private individuals.

[74] *And, as mentioned earlier, continue to monitor the overlay at https://trudi.weizenbaum–institut.de/ipfs_analysis.html.*

In cooperation with ProtocolLabs, the developers of IPFS, we were able to follow the development of the overlay, which has evolved significantly in terms of stability and network health since the crawls in Nov. 2019. Nevertheless, our conceptual insights into overlay measurements and limitations thereof on IPFS versions $<$ v0.5 still hold with minor modifications for newer releases. Furthermore, our monitoring and crawling setup was kept up-to-date with current developments of IPFS and can readily be used.

Moving on, we leave the realm of DHT-based content retrieval towards broadcast-based BitSwap content retrieval. In the next sections, we will paint the missing half of the network layer "map" by monitoring data request patterns of the BitSwap content exchange. In particular, we investigate how BitSwap's design enables the monitoring of data access patterns and trends and discuss the associated risks for the privacy of users. We present (1) a passive monitoring methodology for collecting and processing BitSwap data requests of a large share of the network (cf. Section 4.9) and (2) a monitoring setup as an instance of the methodology (cf. Section 4.10). Our system enables us to reveal *who* requested *which* data item *when*, i.e., which nodeID and IP address requested which CID at what timestamp.

Equipped with measurement traces over a nine month period using two spatially diverse monitoring nodes (with 9.68×10^9 data request entries in total), we highlight possible angles for analyses: (1) estimat-

ing the size of the network (in juxtaposition to crawling results), (2) analyzing activity levels and structure, (3) deriving content popularity distributions, and (4) the feasibility of privacy attacks.

We will start with a concise overview of measurement setup (cf. Section 4.9), followed by exemplary results of what is possible with the data set (cf. Section 4.10), concluded by an assessment of privacy risks (cf. Section 4.11).

4.9 MONITORING DATA REQUESTS

Being a decentralized data storage system, IPFS is inherently hard to monitor. Signaling messages and data are exchanged directly between peers, without passing through centrally-controlled infrastructure that could form a natural vantage point. In the beginning of this chapter, we demonstrated how the IPFS *network* — the nodes, the connections between them — can nevertheless be made visible (cf. Sections 4.4 to 4.8). In the following, we present a methodology for monitoring *data-related activity* in IPFS — how many and which nodes request which data. Notably, this also enables the systematic investigation into what kinds of data are stored on IPFS. Providers only return data when asked for the correct CID, so in order to investigate stored content one must first learn about valid CIDs — which can be done by observing data requests.

This section introduces our methodology for collecting, processing, and interpreting BitSwap data requests. In Section 4.7 we apply our methodology for conducting an exemplary measurement study on IPFS that highlights the feasibility and potential of our approach.

4.9.1 *Data Collection*

Data collection can conceptually be described as a two-step process, with each step leveraging different key features of IPFS' design. Firstly, we form *connections to an unbounded number of nodes*. This is possible because, by design, anyone can deploy a node on the IPFS network and the number of connections a node can maintain is only limited by the IPFS software, i.e., not on a protocol level. Secondly, we *collect all BitSwap messages* from connected nodes. As discussed in Section 4.3, nodes send data requests to all nodes they are connected to, and hence also to our monitoring nodes.

Collecting the BitSwap traffic of a peer allows learning which CID `c` was requested at what time. The monitoring node produces, using a modified version of the IPFS software, a list of (`timestamp`, `node_ID`, `address`, `request_type`, `CID`) tuples. No knowledge is gained about whether (1) the data `d` referenced by `c` was downloaded successfully, and (2) what `d` is (including whether it is a file or a directory). The latter can be determined by downloading

and indexing c. The former can be determined if we also control a provider for the data referenced by c and the data was downloaded from that provider. Alternatively, if the requesting node sends a `CANCEL` at some point, we can afterwards probe its cache by sending it a request for c.

In the presented approach, monitoring nodes are *passive*. They accept all incoming connections, but do not actively search for or connect to peers. They thereby remain indistinguishable from regular nodes in terms of connection initiation and generally do not send BitSwap requests or data. In order to collect messages from a larger portion of the IPFS network, multiple monitors with different node IDs can be used in conjunction.

Our collection methodology implies a number of limitations. While enabling low-cost and hard-to-detect monitoring, passive monitors will generally only detect requests for root hashes of a Merkle DAG. Requests for CIDs further down in the Merkle DAG are sent only to peers in that CID's session, i.e., peers that responded to the initial request (cf. Section 4.3.6). Since our monitoring nodes W do not provide any data, they are not added to any session and will, therefore, not receive any further requests. Furthermore, as outlined in Section 4.3.7, IPFS caches downloaded data. Subsequent request for the same data will be served from the local cache instead of broadcast via BitSwap. We thus only see first requests for data from a peer, or requests after their cache was purged.

4.9.2 *Preprocessing*

Each monitoring node produces a *trace* of BitSwap messages it received. If necessary, traces from multiple monitors can be unified into one global trace. When a node is connected to multiple monitors, we receive broadcast `want_list` entries multiple times. To filter out these duplicates, we consider `want_list` entries received by different monitors to be identical if their source (node ID and address), request type, and target CID match and their timestamps differ by at most 5 s. The window size of 5 s was chosen to account for most genuine duplicates, potentially delayed due to high latencies. Larger values lead to a higher chance of misclassifying regular re-broadcasts of `want_list` entries as duplicates.

As outlined in Section 4.3.6.1, IPFS clients re-broadcast `want_list` entries every 30 s, using an independent timer per connected peer. These repeated broadcasts make up a significant portion of all requests ($> 50\,\%$ according to our measurements), skewing the numbers for some analyses. We mark messages with identical source, request type, and target CID as re-broadcasts if they were received within 31 s from each other. Note that, as nodes maintain independent re-broadcast timers for each connected peer, re-broadcast messages reach different

monitors at shifted times. Depending on the order of preprocessing steps, this can lead to a misclassification of same-monitor re-broadcasts as inter-monitor duplicates.

After data processing, we operate on a unified trace of (`timestamp`, `node_ID`, `address`, `request_type`, `CID`, `flags`) tuples, where the `flags` encode information about duplicate status and repeated broadcast detection.

4.9.3 *Estimating the Network's Size*

The data we gather through our monitoring methodology can be used for estimating the total size N in terms of number of nodes of the IPFS network. For example, using two monitors `m1` and `m2`, and with the simplifying assumption that each monitoring node selects peer sets P_{m1} and P_{m2} uniformly and independently from the whole node population, we can estimate the size of the network as:

$$N_E = \frac{|P_{m1}| \cdot |P_{m2}|}{|P_{m1} \cap P_{m2}|}, \tag{38}$$

where N_E is an estimation for N. This can be derived by considering the population of N nodes as black balls in an urn, $K := |P_{m1}|$ of which are turned red through connections through `m1`. Then, the connections of `m2` can be seen as sampling $n := |P_{m2}|$ balls without replacement — yielding a hypergeometric distribution with $k := |P_{m1} \cap P_{m2}|$ successes. In other words, all parameters of the distribution except the population size N are known, so we can leverage this knowledge of parameters to perform a maximum likelihood estimate of N. If a random variable X follows a hypergeometric distribution, its probability mass function (PMF) is given by:

$$f_X(k|N, K, n) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}. \tag{39}$$

We define the likelihood function as $L(N|K, k, n) := f(k|N, K, n)$. Recall that to obtain a maximum likelihood estimation of N, we have to differentiate the log-likelihood function. Using the Stirling approximation ($\ln n! = n \ln n - n$) this yields

$$\frac{d \ln L}{dN} = \ln\left[\frac{(N-n)(N-k)}{N(N-K-n+k)}\right] \stackrel{!}{=} 0 \tag{40}$$

$$\iff \frac{(N-n)(N-k)}{N(N-K-n+k)} = 1 \tag{41}$$

$$\iff N = \frac{nK}{k} \tag{42}$$

$$\Rightarrow N_E = \frac{|P_{m1}| \cdot |P_{m2}|}{|P_{m1} \cap P_{m2}|}. \tag{43}$$

The general case of r monitors can be handled through modeling the system as a coupon collectors problem with group drawings, also referred to as committee occupancy problem [101, 167, 242]. In this setting, peers correspond to cards and are numbered $1, \dots, N$. Assume each monitor has w connections, then a monitor is a group drawing of w cards/peers without replacement from the total set of peers — hence, there are no duplicates within a drawing but only between them. We furthermore assume these drawings to be independent from one another. Typically, the question modeled is "What is the probability that we have m distinct cards/peers after r draws of size w?". Let X be the number of distinct peers after r draws of size w, then this probability is given by [167]:

$$\mathbb{P}[X = m] = \binom{N}{w}^{-r} \binom{N}{m} \sum_{k=w}^{m} (-1)^{m-k} \binom{m}{k} \binom{k}{w}^{r}. \tag{44}$$

In our setting, we know m (the size of union over all monitors), and N is the quantity to be estimated. Hence, similarly as for Equation (38), we can turn the probability density into a maximum likelihood estimation of N. Differentiating the log-likelihood function (again, with the help of the Stirling approximation), yields a non-closed-form solution for N, given m, r, w, that has to be solved numerically:

$$N - N\sqrt[r]{1 - \frac{m}{N}} - w = 0. \tag{45}$$

The accuracy of our estimation formulas is influenced by a number of factors. For one, due to Kademlia, the peer selection is not uniform but skewed. For example, nodes with node IDs close (in XOR metric) to the node ID of `m1` are more likely to be connected to `m1` than nodes further away. Additionally, we do not fully consider effects relating to nodes' peering "capacity" — a node already connected to `m1` has one less slot for forming new connections and is therefore minimally less likely to be connected to `m2` as well. As IPFS nodes establish anywhere between 600 and 900 connections on average, with even higher peaks, the effect of occupied capacities is probably negligible. With respect to our > 2 monitors estimator, it must be pointed out that monitors actually have a different number of connections at a given point in time, i.e., w is not the same for all monitors. However, a heterogeneous w increases the complexity of the modeling significantly.

In Section 4.10.3 we apply our estimators in practice and empirically compare the result with alternative indicators for the size of the network.

4.9.4 *Content Popularity*

Collected traces of BitSwap requests can be used to deduce the relative popularity of CIDs, and hence the content they reference. Knowledge of this popularity distribution is, e.g., an important building block for the formal analysis of cache hit ratios (especially relevant for IPFS gateways) [108]. It furthermore allows for more realistic network simulations and user models. To this end, we define two different popularity scores, one for capturing IPFS' behavior "on the wire" and one for approximating user behavior. For the former, we define a CID's *raw request popularity* (RRP) as the total number of requests received for a particular CID during a given period. This number is of interest for simulation studies and for improving the performance of BitSwap. For approximating user behavior, we consider *unique request popularity* (URP), the number of distinct peers that requested a respective CID in the given time period. The motivation behind URP is that requests for a CID coming from distinct peers indicate the corresponding data item's popularity among distinct users.

4.10 EXAMPLE MONITORING STUDY

We implemented our monitoring methodology and collected data from the public IPFS network from March 2020 until February 2021. In the following, we describe our setup and present exemplary observations that it has allowed us to make, showcasing the feasibility and utility of our approach.

4.10.1 *Monitoring Setup*

Our setup consists of a small number of monitoring nodes W and an additional larger set of *resolving nodes* R. The nodes W function as outlined in Section 4.9.1. We use the nodes R for actively resolving, downloading and indexing content discovered through W.

The goal of W is to gather traces for as many peers as possible. To that end, they accept all incoming connections and impose no limit on the number of connections they maintain. They are furthermore deployed on publicly reachable servers, i.e., not behind a NAT. For this measurement study, $|W| = r = 2$, with one node situated in Germany and one in the United States. The nodes were running a modified version of the IPFS client v0.5.0-rc2 for the majority of the duration of data collection. One node was later upgraded to IPFS v0.7 in late October 2020.

[75] *The corresponding tools are maintained at https://github.com/-mrdoll4r/ipfs–resolver*

The nodes R download and analyze content discovered via W's logging, using unmodified IPFS clients[75]. For any CID c recorded by W, the resolvers R try to resolve c to its respective block b and all blocks in the Merkle DAG below b. Hence, although we do not

actively enter a peer's session for a data request, by resolving CIDs and traversing their Merkle DAG we can extrapolate which content was requested in the sessions. The nodes in R run on different machines, in different locations around the globe, using different ISPs and unique node IDs, to both rule out any network restrictions as well as being able to find content with a high probability, if it is available on IPFS. CIDs that could not be downloaded initially are recorded and the downloads are retried at a later date.

4.10.2 *Data Collection*

We collected BitSwap traces continuously for nine months, yielding over 500 GB of compressed traces. We maintained only one monitoring node for the first two months and a total of two for the remaining seven months. Since beginning the data collection, our monitors underwent minor configuration and version changes as well as some outages. We classify these as minor (>5 s) and major (>1 h) outages and count the number of days during which they occurred. The `us` monitor in the U.S. was running for a total of 285 days of which 6 days experienced at least one major outage and an additional 13 days experienced at least one minor outage. The `de` monitor in Germany was running for a total of 224 days of which 7 days experienced at least one major outage and an additional 4 days experienced at least one minor outage. We observed more than 510 million unique CIDs, more than 350 million of which we were able to download and index. For indexed CIDs, we store the metadata and links contained in the respective blocks but discarded the data itself due to its immense size.

The remainder of this section showcases analyses possible with our collected data, and with data collectable using our monitoring methodology in general. We present results for:

- Estimating the size of the IPFS network.
- Assessing the level and structure of data-related activity.
- Investigating the popularity of content stored on IPFS.

We leave further analyses on the "file system" layer of IPFS that are enabled through our methodology, e.g., of deduplication and the structure of IPFS' data graph, to future works.

4.10.3 *Monitoring Coverage and Network Size*

In the following, we take a closer look at traces collected in the week from July 30th to August 6th, 2020. Over this period, our two monitors saw 41647 and 41151 unique peers in total, respectively, for a union of 45115 peers. The monitors were connected to an average number of,

respectively, 6600 and 7000 peers, with the size of the union of unique peers being 9300 on average. Notably, averages and weekly totals differ significantly from each other, which is in line with previous observations about churn in the IPFS network [6].

The collected data points allow us to apply the network size estimation formula we propose in Section 4.9.3. Doing so yields an estimated average network size of 10811 with Equation (38) and 10800 with Equation (45) (setting w as the average of connections of both monitors). While deriving a ground truth for this estimate is inherently challenging [149], we can compare our results with alternative indicators for the IPFS network's size. Existing measurement infrastructure from previous studies (cf. Sections 4.3 to 4.8) gives us insights into the portion of the network reachable through crawls of the IPFS DHT[76]. Crawls of the IPFS network during the discussed period revealed a total of 36682 unique peers, with an average network size of 15900 peers per crawl. This hints at the fact that our method might underestimate the current network size. However, measuring the size of the IPFS network based on DHT crawls has limitations on its own. For example, crawled IPFS nodes also propose DHT nodes to our crawler that are in fact offline [220]. Such nodes are still counted by the crawler, as even online nodes might be unreachable to it if they reside behind NAT or other restrictive network devices. On the other hand, our DHT crawler doesn't enumerate client-only nodes that are part of the IPFS network but not part of the DHT, which explains why our monitors saw more unique node IDs over the discussed week than our crawler. While a detailed evaluation of different network size estimation approaches is out of the scope of this work and is left for future investigations, the different estimates give us a ballpark assessment of the number of nodes in the IPFS network.

[76] *See also the aforementioned https://trudi.-weizenbaum–institut.de/ipfs_-analysis.html.*

Given estimations for the network's size, we can now gauge the *coverage* of our monitoring approach. We use the crawling-based estimation of the network's size in the following, being the larger of the two and therefore more likely to underestimate our coverage. At any given time, our monitoring nodes `us` and `de` were thereby connected to, and hence receiving BitSwap messages from, 44 % and 41 % of the network, respectively. The joint setup combining traces from both nodes had an average monitoring coverage of 58 %. Notably, we achieved this coverage using only two passive monitoring nodes. The monitoring coverage can be further increased by adding more monitoring nodes or, complementary, by implementing a more active, crawler-like (and perhaps targeted; cf. Section 4.11) peer discovery mechanism.

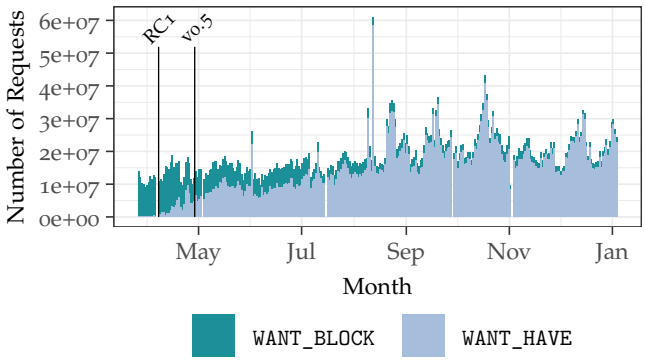


Figure 38: Transition of request types over time, from `WANT_BLOCK` to the newer `WANT_HAVE`, indicated by the number of requested entries per type. X-axis ticks mark the beginning of the indicated month.

4.10.4 *BitSwap Usage*

Only a small portion of the nodes we monitored in the above period were actively engaging in the BitSwap protocol, sending at least one request or cancel. The two monitors saw 1657 and 1685 unique BitSwap-active peers over the duration, respectively, with a union of 2073 unique BitSwap-active peers. Still, we observe a growing BitSwap (and hence IPFS) usage between March 2020 and January 2021, measured by the number of BitSwap data requests per day. Figure 38 depicts the view of monitor `us` on the number of requested CIDs per day and entry type, with the two entry types stacked vertically. Missing values indicate incomplete data due to outages. Annotated with black lines are release dates for `go-ipfs` versions. We observe an upwards trend in the total number of requested entries over the entire duration, as well as a shift from `WANT_BLOCK` to `WANT_HAVE`. The `WANT_HAVE` entry type was introduced with `go-ipfs` v0.5 (respectively, in release candidates for that version), and we see its adoption around that time. This shows a willingness of users to upgrade their clients. The large spike at the beginning of August was registered by both of our nodes, but we did not investigate further.

We also analyzed the collected CIDs for the Multicodec they reference. The Multicodec describes what type of data is referenced by a CID, as outlined in Section 4.3.3. The results of our analysis over the entire $\approx$ 9 months are presented in Table 7. Over this duration, we collected a total of 9.68×10^9 data requests in sum over both monitors and calculate the share of these. We only count requested entries, not `CANCEL`s, and derive the data from raw, unprocessed traces of the two monitors. We can see that, from the perspective of our monitoring infrastructure, IPFS is used mostly for file storage.

Codec	Share (%)
DagProtobuf	84.86
Raw	14.42
DagCBOR	0.70
GitRaw	0.01
EthereumTx	< 0.01
Others (8)	< 0.01

Table 7: Share of data requests by Multicodec. (March 2020–January 2021)

Country	Share (%)
NL	43.5
US	34.9
SG	9.0
DE	6.9
CA	1.6
Others	< 4.0

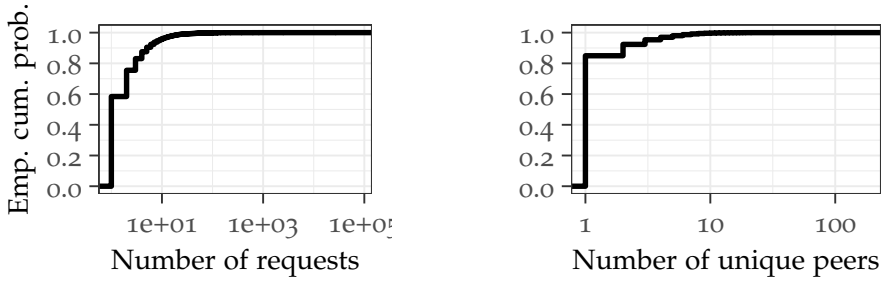
Table 8: Share of data requests by country. (July 30th–August 6th, 2020)

Our collected traces also allow insights into the geographic patterns of IPFS usage. We examined IP addresses from our unified, deduplicated dataset over the period between July 30th and August 6th, 2020 and resolved them, as in Section 4.7, via the MaxMind GeoIP2 database. Table 8 shows the share of observed BitSwap data requests per origin country. We see that nodes residing in the Netherlands and the US together account for roughly 78 % of all observed activity during this period.

4.10.5 *Content Popularity*

Applying the popularity scores we defined in Section 4.9.4 to the collected traces for the July 30th to August 6th period allows us to calculate the distribution of CID popularities. We compute the unique request popularity (URP) on the unified traces of both monitors and the raw request popularity (RRP) on the trace of the us monitor which captures the raw request distribution arriving at one node. We present the resulting empirical CDFs (ECDF) in Figure 39. It can be seen that the majority of CIDs in both distributions have a low popularity score, in particular, over 80 % of CIDs were only requested by one peer, as depicted in Figure 39b. Although the distributions differ, both seem highly skewed with few highly-requested CIDs and a majority of "unpopular" ones. In contrast to other works on item popularity reporting heavy-tailed, Zipf-like distributions [108], fitting a power-law distribution to our measured scores (Figure 39) as laid out in [59] (and already showcased in the analysis of Figures 33 and 34) yields a p-value $\ll 0.1$, both for RRP and URP, regardless of the choice for a cut-off value x_{min}, with 2500 bootstrapping runs. We therefore conclude that the power-law hypothesis has to be rejected, i.e., that the measured popularity data does likely *not* follow a power-law distribution.

It has to be noted that popular data items according to RRP are often not resolvable, i.e., the data block the CIDs are pointing to cannot



(a) RRP score (b) URP score

Figure 39: ECDFs of content popularities. (July 30th–August 6th, 2020)

be found. This observation may stem from different factors. First, BitSwap periodically re-broadcasts requests for CIDs it cannot resolve (cf. Section 4.9.2). Furthermore, some peers issue an unexpectedly high number of requests for the same data item — hinting at configuration errors or non-standard usages of IPFS.

The ten most popular CIDs according to URP are resolvable. However, many of these most popular CIDs contain faulty content that we were not able to extract, i.e., the respective data block can be obtained from the network but cannot be parsed due to errors. The most requested syntactically valid content according to URP stems from the dAppNode-project[77], a project devoted to simplifying the process of running and maintaining full nodes in various P2P networks.

[77] https://github.com/dappnode/DAppNode

4.11 PRIVACY RISKS

The monitoring of data requests in IPFS provides useful insights for tuning the network's performance, assessing its level of use, and identifying key usage scenarios. However, our monitoring techniques can also be used for tracking the behavior of individual users, implying latent privacy risks. In the following, we flesh out these risks by proposing a series of specific attacks on the privacy of IPFS users. As a demonstration, we uncover the (normally hidden) node identifiers corresponding to public IPFS/HTTP gateways and successfully track requests initiated through these gateways. We also discuss countermeasures, highlighting that the identified attacks are in part enabled by core aspects of IPFS's design, and that remedying them is a hard challenge when other desirables from a decentralized data storage system are taken into account.

4.11.1 *Privacy attacks on IPFS*

We define three specific attack vectors that can enable adversaries to learn about the current and past data request behavior of IPFS nodes: *Identifying Data Wanters* (IDW), *Tracking Node Wants* (TNW), and

Testing for Past Interests (TPI). In the standard usage mode, IPFS users access and distribute data via IPFS nodes under their own control. Consequently, learning what data a node is or has been interested in allows for direct conclusions about the (likely private) preferences of its human operator[78]. We discuss the alternative, (public) gateway-based usage mode of IPFS in Section 4.11.3.

[78] *Note that even if data items have been encrypted before being placed on IPFS, metadata such as request behaviors and approximate data sizes can still be learned by an adversary.*

4.11.1.1 *Identifying Data Wanters (IDW)*

The goal of the IDW attack is to discover nodes that are interested in a given CID-identified data item. The setup of the attack is identical to our monitoring setup. In fact, our deployed monitoring infrastructure (cf. Section 4.10.1) already collects the necessary information for listing node IDs that have requested a given CID. The effectiveness of the attack (in the sense of the percentage of captured CID requests) is proportional to the invested resources. By deploying more monitoring nodes or using an active, crawling-like peer discovery approach, the adversary can increase the number of nodes to which he maintains a direct connection and from which he receives CID messages. Already with one monitoring node, however, we were able to monitor more than 40 % of the public IPFS network (cf. Section 4.10.3).

4.11.1.2 *Tracking Node Wants (TNW)*

The TNW attack revolves around tracking which data items a given target node is interested in. It is therefore sufficient for the adversary to maintain a connection to the target node and collect the requests that the target node broadcasts. From a practical standpoint, a more challenging aspect of the TNW attack is to determine the node ID or IP address of the target node, as the adversary goal will likely be the surveillance of the user or organization *behind* the node. While different more general ways are conceivable to learn the IP address of a victim, this is also possible by again leveraging IPFS itself: the IDW attack can be used to discover nodes that are requesting some CID only the victim is likely to know or be interested in (e.g., because the adversary sent it to him). In Section 4.11.2, we demonstrate the effectiveness of this approach on well-known public gateway nodes on the IPFS network.

4.11.1.3 *Testing for Past Interests (TPI)*

In the TPI attack, the adversary seeks to confirm that a given node has recently accessed a given data item. The attack leverages the fact that IPFS nodes *cache* previously requested data items locally and *serve* them to interested parties (cf. Section 4.3.7). This caching mechanism is a cornerstone to the scalability and censorship-resistance of IPFS. However, in its current form it also enables any adversary capable of joining the IPFS network to test whether a given target node has

previously requested a given CID — by sending a CID request to the target node. The target node will answer if the sought data item is in its cache, and the data item will only be in the target node's cache if it was either requested or initially uploaded via the target node based on a user request. Like for the TNW attack, the TPI attack can be mounted with negligible resources required by an adversary once the IP address or node ID of the victim becomes known.

4.11.2 *Proof of Concept: Tracking Gateway Requests*

IPFS offers a bridge to access the IPFS network and hosted content through HTTP. While every node offers this translation locally by default, there are also a number of *public gateways* available on the regular HTTP-based web, maintained by the developer community and a number of organizations. Public gateways are a convenient way to access IPFS and aim to boost adoption of IPFS, showcase examples, and enable access to the network in situations when running nodes locally is infeasible. Public gateways also allow us to test privacy attacks in a realistic setting without threatening users. In the following, we outline a methodology that combines the IDW attack with a tailored probing step for linking well-known public gateway nodes (identified by their DNS names and IP addresses) to IPFS nodes (with node IDs and often different IP addresses than the associated HTTP endpoints). We apply the TNW attack on the identified nodes and briefly discuss the results of this investigation, which might be of independent interest. The success of the presented experiment underlines the viability of the proposed attacks and the latent privacy risks they imply for IPFS users.

4.11.2.1 *Gateway Probing Methodology*

We build upon our monitoring methodology (Section 4.9), extending it by a *probing* step. We generate a unique block of random data, yielding a unique CID c[79]. We then launch a IDW attack on c, adding our monitoring nodes W as providers for c to the IPFS DHT. Subsequently, we ourselves request c through the HTTP-facing side of a public gateway and wait for BitSwap messages to arrive. If nodes in W are currently connected to the gateway, this usually happens within seconds. Without a connection, we can rely on the gateway to find a member of W through the DHT, and connect our nodes after that. Once found and connected to W, the gateway will issue a `want_list` update containing the CID c. The received BitSwap request for c enables us to uniquely identify the IPFS side of the probed gateway. Since c refers to a block of randomly generated data, it is unlikely that any other user on the network would request c, yielding a large confidence in the measured results.

[79] *Obtaining a CID duplicate is improbable due to the fact that CIDs are based on cryptographic hashes of the data (s.a. Section 4.3).*

4.11.2.2 *Gateway Probing Results*

We applied our gateway probing method to a list of 38 public gateways curated by the IPFS project [219]. We repeated the measurements two times, on the 31st of May and 8th of June 2020 and from two distinct hosts situated in Germany and the U.S., respectively. For each date and each gateway we used a different random seed so that c is unique in each trial. The complete results as well as a replication of the list of public gateways at the respective dates are gathered in the Appendix, cf. Table 16. 24 of the gateways in above mentioned list were functioning properly at the time of our tests, while 12 of the 38 were not usable, in line with the public list [219]. Using our probing method, we were able to discover node IDs for all 24 functional public gateways, receiving relevant BitSwap messages from them at least once. We also discovered node IDs for two of the non-functional gateways, i.e., our HTTP requests to them did not succeed but we still received relevant BitSwap messages. We suspect a misconfiguration on the HTTP end of those gateways.

After collecting data through our probing step, we cross-referenced the discovered IPs and overlay IDs with peer lists from our monitoring nodes W, focusing on discovering nodes that share IP addresses and node IDs associated with multiple IP addresses. The probing and cross-referencing uncovered that several public gateways in above list were in fact associated with not just one but multiple nodes on the IPFS network. We contacted the operators of one prominent gateway with 9 associated IPFS nodes, who confirmed that, to their surprise, we correctly identified all of their nodes. In total, we discovered 44 node IDs corresponding to well-known public gateways.

4.11.2.3 *Public Gateway Requests*

After successfully discovering the IPFS nodes associated with well-known public gateways, we can launch a TNW attack on these gateways. In the following, we discuss results collected from July 30th to August 6th. Figure 40 depicts the number of BitSwap requests per second that our two monitors received during this period. We unified and deduplicated the traces from the two monitors as per Section 4.9.2. We plot both gateway and non-gateway ("homegrown") request rates, illustrating that we can successfully track the requests sent by a target node population. We find that a significant portion of gateway traffic is due to a single public gateway operator, Cloudflare, and therefore mark Cloudflare traffic explicitly in our results.

Our monitors were engaged in regular monitoring (as per Section 4.9) during this experiment and thereby collected BitSwap messages from non-targeted nodes as well. As can be seen in Figure 40, we in fact received far more requests from non-gateway nodes than from gateway nodes. This observation might indicate that gateway *usage* is

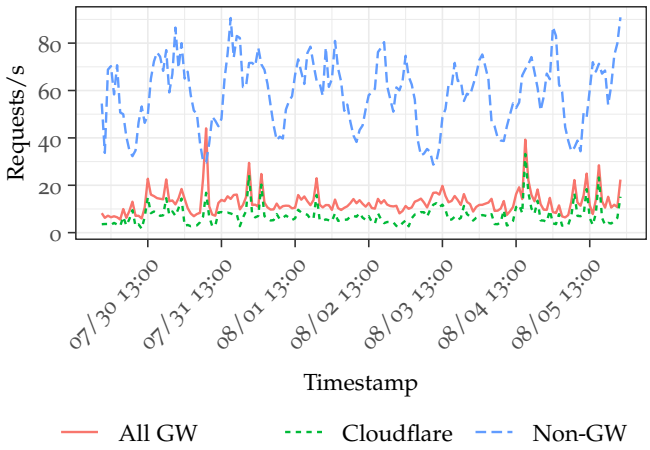


Figure 40: Deduplicated BitSwap request rate by origin group. The rate is calculated over one-hour slices of the unified, deduplicated trace as explained in Section 4.9.2. Times are in UTC.

similarly only a small part of total IPFS usage. However, we must expect that public gateways cache requested data at least as aggressively as regular nodes — if data requested via HTTP is already cached, no BitSwap request is generated and we thus cannot learn of this request. Operational data from public gateway operators, for example about observed cache hit ratios, might enable more conclusive insights w.r.t. the popularity of public gateway services.

4.11.3 *Privacy enhancement for decentralized data storage systems*

The feasibility and relevance of the IDW, TNW and TPI attacks are due to a number of inherent characteristics of IPFS:

1. Long-lasting node identifiers — as connections must be maintained, nodes retain their node ID and IP address(es) for extended periods of time.

2. No connection limits — there are no mechanisms in place that can reliably prevent a small set of (adversary or monitoring) nodes from maintaining connections with nearly all nodes in the network, or a single adversary node from connecting to specific victim nodes.

3. CID request broadcasts — as part of the BitSwap protocol, nodes broadcast data requests to all nodes they are connected to.

4. Cooperative caching — nodes cache downloaded data and cooperatively serve it to other interested nodes.

5. Single-user nodes with no cover traffic — users accessing the IPFS network via a locally installed IPFS client are represented in the network as a node that relays only their own and actual requests.

The weakening of these attack enablers poses a significant challenge for designers of decentralized data storage systems. Naive countermeasures can easily result in a significant deterioration of performance, censorship-resistance or peer-to-peer scalability. For example:

1. Node identifiers can be cycled more frequently and an additional anonymization layer for IP addresses can be used. The effective cycling of node identifiers (i.e., existing connections need to be teared down) essentially increases churn. Using an established IP address anonymization service like Tor [254] limits the performance of the data storage system to that of the service. It is an open question how to best integrate IP address anonymization functionality into IPFS itself and what the performance impact of such a change would be. Parallels can likely be drawn to the design and operation experience of privacy-centric systems such as Freenet [58, 229].

2. Systems like IPFS thrive on their openness, on allowing anyone to join the network and provide a node. Attempts to limit the amounts of connections a single node can maintain are difficult to design and limited in effect as adversaries can easily split their connections between multiple Sybil [86] nodes. Introducing per-connection *costs*, e.g., by requiring continuous Hashcash-like [20] proofs of work from peers, will likely also result in a decreased population of honest nodes while being of uncertain effect against determined adversaries with access to cloud computing resources.

3. Nodes could request data items only from nodes found via DHT queries, rather than from their whole overlay neighborhood. However, this would reduce IPFS's robustness against censorship attacks (cf. Section 4.3) while being of limited effect as (Sybil) adversaries can also place themselves at key locations in the DHT (e.g., the adversary can generate a node ID such that he is co-responsible for a target CID, thereby being able to track DHT lookups for that CID). On the opposite end, extending BitSwap to support multi-hop requests or even flooding might not be sufficient to confuse dedicated adversaries either, as the identification of messages sources is provably feasible even in decentralized flooding-based networks [207].

4. Users can manually remove problematic items from their node's cache. While this is definitely helpful against a time-delayed TPI

attack, it requires manual diligence and has no effect on IDW and TNW attacks. More nondiscriminatory cache pruning would directly deteriorate censorship resistance and overall performance by reducing the number of data copies in the network.

5. Adding realistic cover traffic to add plausible deniability to one's genuine data requests is hard — in order for fake data requests to be effective they must be directed an actually existing CIDs and follow realistic popularity distributions. Lists of existing CIDs and their popularity distribution might be obtainable for monitoring operators (cf. Section 4.10.5), but is not usually the case for regular users.

Users could also use IPFS via a public gateway (cf. Section 4.11.2). While this measure is both highly effective in terms of privacy-protection (users can additionally access the gateway via an anonymization service such as Tor) and already in active use by at least part of the IPFS user base (based on our monitoring results and numerous anecdotal evidence), it arguably weakens the benefits of IPFS as a *decentralized* data storage system. Namely, by accessing content on IPFS without also forming a node of the IPFS network users do not contribute to the network's scalability and censorship resistance.

Turning the gateway strategy on its head, regular IPFS users could also *provide* gateway services, thereby both strengthening the IPFS network and adding natural cover traffic to requests they themselves send via their node. For increasing the use of smaller gateway services, changes to the IPFS software are necessary so that the selection of a gateway is handled automatically. The solution can be expanded towards a form of onion routing [254], with data requests being routed through three or more gateways and only the last gateway in the chain performing an actual DHT- and BitSwap-based search, or a more lightweight deniability-increasing approach like *AP3* [197].

We leave the in-depth investigation of this and other potentially promising privacy-hardening approaches for future works. We also note that providing gateway services to other (anonymous) users might imply some legal uncertainties. Gateway nodes can be led into caching data the mere possession of which might be considered a criminal act. Depending on jurisdiction, providers of hosting services are often exempt from direct liability as long as they quickly remove problematic data upon obtaining knowledge or awareness of its existence (see for example [94, Art. 14]) and it is plausible (albeit, to the best of our knowledge, not certain) that such safe harbor rulings would extend to small-scale IPFS gateway nodes.

4.12 CHAPTER SUMMARY

4.12.1 *Summary*

In this chapter we provided a comprehensive, empirically-based perspective on network and content aspects of IPFS. We have seen that extensive network layer monitoring of IPFS is possible through (1) crawling the DHT which yields a subset of established connections between peers, as well as (2) monitoring BitSwap content requests, enabling us to paint an in-depth and holistic picture of IPFS' adoption, usage patterns and network health.

Regarding the DHT, we conducted a seven day crawl of the IPFS DHT in Nov. 2019 (C_1) and a two week crawl in Feb. 2021 (C_2), enabling insights about node population as well as indicators about the larger overlay structure. In C_1 we found a total of $4.4 \cdot 10^4$ on average, 6.6 % responded to our connection attempts, as symmetric NATs were not properly handled by IPFS at the time. Since Nov. 2019, the number of unreachable nodes in the network has decreased significantly, with $1.1 \cdot 10^4$ nodes found in Feb. 2021, almost 50 % of which could be dialed by our ipfs_crawler. Several indicators hint at the fact that a majority of nodes is operated by private individuals, e.g., short session lengths (for C_1) and daily periodic fluctuations in the number of nodes in the network. Interestingly, especially for C_2, the overlay graph is not scale-free and robust against random outages as well as targeted attacks — resembling results on Gnutella [250] instead of other Kademlia systems [232].

We published, maintained and improved the code of ipfs_crawler in cooperation with ProtocolLabs — the designers of IPFS and the libp2p networking library[80]. Furthermore, we made the tools and evaluation scripts for processing crawl data public and easy to use, so that other researchers can replicate our study and compare results.

[80] *https://github.com/scriptkitty/ipfs-crawler.*

Regarding the monitoring of BitSwap requests, we presented a methodology for observing data-related activity in IPFS of large portions of IPFS' node population at low cost (more than 40 % with one passive monitoring node). Our two monitoring nodes received indicators for all CIDs that connected-to nodes request and look for. We tested our methodology[81] in a 9-month measurement study and highlighted several angles for analyses. For example, we find that data-related activity was increasing over time. For a studied 7-day period, almost 80 % of activity can be attributed to just two countries and the popularity of content items does not follow a power-law distribution. Despite its potential for generating informative insights about the IPFS network, the effectiveness of our monitoring approach also paints a troubling picture with regard to user privacy. We described how our methodology can be extended for realizing a range of privacy attacks that ultimately enable low-resource adversaries to surveil the current

[81] *Again, published and maintained at https://github.com/mrdoll4r/ipfs-resolver.*

and past data requesting behavior of node-operating users. While not covered in this chapter, the methodology and dataset enables, among other things, in-depth investigations into the filesystem layer of IPFS and the content stored on the network.

4.12.2 *Conclusion*

Given the diverse analyses and datasets gathered on IPFS through our methodologies, what can be said with regards to the research questions on IPFS from Section 4.1?

We have studied two potential avenues for monitoring and measurements. Although limitations exists in the form of non-crawlable connections, the constraint to root hashes in monitored BitSwap requests and caching, the combination of both approaches allows for painting a holistic picture of IPFS.

[82] *https://github.com/-libp2p/hydra-booster.*

Although there are centralization tendencies, e.g., through infrastructural nodes such as IPFS/HTTP-gateways and DHT "boosters"[82], a significant share of IPFS nodes seems to be operated by private individuals. Having studied IPFS for more than a year, we were able to observe an increase in maturity and performance, but also in centralization. For example, a large share of active participants in the DHT are boosters[83], specifically designed to bolster the performance of DHT lookups. This evolution is clearly visible when comparing the two sets of crawls C_1 and C_2 from Nov. 2019 and Feb. 2021, respectively. The overlay topology suggest that IPFS' overlay bears some resemblance to scale-free and small-world networks, more so for C_1 than for C_2. Interestingly, the degree distribution does not necessarily follow a power-law for both sets of crawls, though the difference is more noticeable in C_2 which has a surprisingly homogeneous degree distribution. Subsequently, this renders graphs from C_1 more vulnerable to targeted node removals than graph from C_2. The differences between the two sets of crawls potentially stem from significant improvements of the libp2p networking library and our crawler as well as higher number of long-running infrastructural nodes.

[83] *The largest share of nodes seems to stem from the "Storm"-Botnet which uses IPFS for command & control [224], which started to emerge in 2020.*

The unusual combination of an structured overlay as well as unstructured broadcasting for retrieving content makes IPFS robust against adversaries, e.g., in the form of Sybil/eclipse attacks. Nevertheless possible [220], it requires significant resources by an adversary and is not guaranteed to be successful. Although this resilience comes at the cost of performance (in particular bandwidth), it is a desirable property to have, especially in the context of blockchain applications which have to be particularly robust against attacks. However, while contributing to additional resilience, the privacy of IPFS users is severely impaired. Not only is it possible to paint a complete picture of which nodeID/IP address has requested what content at what time by simply monitoring data requests; through active polling even more information

can be extracted, e.g., which peers hold and serve which data items. Regarding individual user's privacy, this is clearly undesirable, as it enables tracking not only by "individual" adversaries but also state actors. Depending on a node's jurisdiction, this is a two-edged sword. One the one hand, these tracking capabilities are useful for thwarting the distribution of illegal content such as child pornography (which we found and reported to the German law-enforcement authorities in the course of our studies). On the other hand, it can easily be abused to persecute node operators sharing information that contests an oppressive regime.

Extracting which nodes hold specific content can also be potentially dangerous in blockchain applications. An adversary could employ a monitoring setup similar to ours in combination with active polling to obtain which nodes host the content that the application is relying on. Depending on the degree of replication, the attacker might be able to take down exactly those nodes in targeted attacks to effectively block the respective content. Furthermore, while IPFS itself is moderately decentralized as we have seen, the linkage between, e.g., Ethereum and IPFS is often carried out through centralized services like Pinata [83] — effectively questioning the benefits of IPFS in the first place.

Moving on from IPFS as one popular system for distributed storage (whose potential far exceeds blockchain systems), we turn our attention the last building block of this thesis: monetary stabilization in cryptocurrencies.

5

MONETARY STABILIZATION IN CRYPTOCURRENCIES

5.1 OVERVIEW

Having studied Ethereum on the infrastructural layer and IPFS as one popular representative system for distributed storage, we shift our focus from a purely overlay-based perspective on decentralized systems and their robustness towards a hybrid between computer science and economics. This shift perspective is necessary to study the final building block (cf. Section 2.6) of this thesis: monetary stabilization in cryptocurrencies in the form of so-called "stablecoins". These cryptocurrencies promise the best of two worlds: a (permissionless) system such as Bitcoin combined with the price stability of traditional fiat currencies such as the US Dollar. The desire for cryptocurrencies with stable value is mainly motivated by the large fluctuations in coin prices of ordinary cryptocurrencies like Bitcoin and Ethereum. Their volatility has often been cited as one of the reasons for every day user's reluctance to adopt Bitcoin and the like as means of payment [180, 231, 276]. This statement can be further underlined by the popularity of existing stablecoins, as two of them are among the 30 largest cryptocurrencies by market capitalization: Tether [193] and Dai [185]. In that, stablecoins aim to fulfill two of the three commonly cited functions of money [276][84]: (1) medium of exchange and (2) store of value, whereas normal cryptocurrencies are only able to fulfill (1). Stablecoins are therefore a fascinating topic of research, as their roots and applications lie in the technical realm of decentralized cryptocurrencies while their goal is inherently economic.

[84] *Medium of exchange, store of value and unit of account*

Stablecoins are of importance for blockchain applications, in that participants in these systems are often incentivized through monetary rewards in the form of cryptocurrencies. Volatility in the value of the reward may therefore lead to misaligned incentives. Additionally, sta-

This chapter is based on previous collaborative work [9]. Similar to the other publications, I conceived the initial idea of classifying existing stablecoin projects which evolved into the taxonomy presented here through discussions with my co-authors during the process of writing the publication. Finding, analyzing and classifying existing projects was predominantly carried out by Ingolf Pernice and Roman Proskalovich. Also, IP and RP as economists contributed the majority of insights from sifting through existing economic literature. My role and contribution was to provide technical counterbalance: a knowledgeable view of economics combined with a computer science background, with the goal of enabling a transfer of knowledge from economics to CS (the audience of [9]). This concerns particularly the monetary regimes (cf. Section 5.7), exchange rate arrangements (cf. Section 5.6) and speculative attacks (cf. Section 5.6.3).

blecoins are one core building block for decentralized finance (DeFi), where especially Dai is used as a foundation for complex derivatives [126, 222, 268]. At the origin, DeFi applications are a consequent progression of Bitcoin's goals — seeking out to replace the traditional financial system with decentralized, transparent and permissionless alternatives. If DeFi adheres to these ideals is subject to debate [126, 268]. Independently of this debate's outcome, one has to acknowledge that DeFi has established itself as an increasingly popular blockchain application which relies on the infrastructural layer (cf. Chapter 3) as well as stablecoins.

In this chapter we will investigate whether stablecoins are able to fulfill their ambitious goal of providing a medium of exchange *and* store of value in a permissionless fashion. In contrast to the preceding chapters, where robustness was measured by purely technical means, the robustness of a stablecoin system is additionally related to their ability to maintain a stable value despite adversarial market forces. To achieve this hybrid perspective of computer science and economics, we develop a comprehensive taxonomy on stabilization approaches for cryptocurrencies. The taxonomy combines insights on monetary theory and stabilization of fiat currencies from existing economic literature with the specificities of cryptocurrencies. In particular, we extracted generalized design features from existing stablecoin concepts[85] to capture the (im)possibilities of cryptocurrencies, i.e., not all techniques from the fiat world are applicable to stablecoins and vice versa. Our taxonomy[86] tackles three broad questions that are reflected in the structure of this chapter:

1. Which types of practical techniques are used to achieve stability? (cf. Section 5.4)
2. In what way can the value of a cryptocurrency be linked to that of an external currency (e.g., the US Dollar (USD), the Euro (EUR))? (cf. Section 5.6)
3. What is the stabilization target (e.g., exchange rate to USD, inflation, etc.)? (cf. Section 5.7)

We use our taxonomy to systematically explore and analyze the stablecoin landscape, allowing us to highlight the current state of development from different dimensions and uncover blank spots. Due to the taxonomy, we are able to go beyond individual projects and instead provide an abstract overview of concepts merged with approaches from traditional monetary policy. This abstract perspective allows us to reason about fundamental properties, risks and limitations of stability techniques in practice, whereas the short-lived nature of most cryptocurrencies quickly would render a mere survey of existing coins obsolete. As our taxonomy bridges computer science and economics it allows for transfer of expertise. This not only leads to the detection of risks but also reveals avenues for future research.

[85] *We analyzed white papers, websites and, when available, price data of 24 stablecoin projects.*

[86] *The complete taxonomy can be found in the Appendix, cf. Section 7.2.2.*

On a more detailed level, applying our framework to a total of 24 stablecoin projects, we find that simple tokenization of national currency is the most popular stabilization technique. This approach involves a 1:1 exchange of cryptocurrency for centrally deposited fiat currency, e.g., employed in popular coins like Tether. More sophisticated techniques have been planned for implementation, however, many face inherent challenges such as not allowing a permanent reduction of the money supply. Furthermore, almost 40 % of surveyed coins promote a problematic combination of exchange rate targeting (i.e., stabilizing the exchange rate of the coin to a fiat currency) and techniques for reducing the coin supply with either limited reserves or a potentially unlimited supply of self-issued tokens. While more research is encouraged, there are indications that this combination might render them vulnerable to speculative attacks, i.e., scenarios in which investors deliberately apply market pressure to push the price of a coin below the stable value to make a profit [145, 146]. Given these risks, the inability of respective coins to maintain a stable value in the past, and insights from existing economic literature, we argue that these so-called soft pegs are not maintainable in the long run. Instead, more sustainable arrangements such as smoothing of short term variations or hard pegs are preferable.

Lastly, almost all analyzed stablecoins rely on a trusted price feed and therefore a functioning decentralized oracle. This assumption is problematic, as existing research [10, 215, 280] does not solve the decentralized oracle problem for arbitrary values, and a general solution might be impossible due to the lack of strong identities [86] or missing incentive-compatibility.

The remainder of this chapter is structured as follows. After discussing related work (cf. Section 5.2), we specify our analysis methodology for building our framework and the classification of stablecoin projects in Section 5.3. We devote a section to each big question in our taxonomy: (1) avenues to achieve stability (cf. Sections 5.4 and 5.5), (2) exchange rate arrangements and stability guarantees (cf. Section 5.6) and (3) stabilization targets in the form of monetary regimes (cf. Section 5.7). In every section we first introduce the theoretical concepts before applying the taxonomy by classifying existing stablecoin projects. Lastly, we touch upon decentralization and trust in the presence of oracles in Section 5.8. Our hybrid perspective between computer science and economics provides a valuable transfer of knowledge and opens up new perspectives on a predominantly technical discussion.

5.2 RELATED WORK

When we wrote the publication this chapter is based on [9], monetary stability in cryptocurrencies had barely been studied by the scientific

community. Since then, a plethora of surveys, classification and taxonomies have been proposed [45, 145, 198], even with one basically identical copy of our work [199]. Despite these recent developments, this related work section will not cover those. The related work at the time of writing of [9], i.e., available papers and concepts, highly influenced our method and the resulting taxonomy, and the reader should be able to follow our considerations. Therefore, to not distort this process, the following paragraphs cover what was available at the time.

Iwamura et al. [137, 231] propose a combination of dynamic mining reward and automatic inflation of coins. In a different approach, Caginalp et al. argue [48] that since cryptocurrencies have no underlying value measured by "traditional techniques" that are used to value stocks, bonds or derivatives, new models are necessary. Following this idea, Caginalp [47] uses asset flow equations to model the price of cryptocurrencies and derive conditions under which the models differential equations stabilize. In contrast to proposing in-depth designs of novel stabilization approaches, we focus on surveying existing projects and outlining principal features of the design space.

Another branch of scientific research concerns itself with *central bank digital currency (CBDC)* [30, 40, 68, 148]. We deliberately chose not to cover this topic, since the central bank, as the central actor, remains in control of both monetary policy and mining. Effectively, this creates another form of national money, leaving monetary policy aspects mostly unchanged.

There already exists variety of (non-scientific) classifications and stablecoin lists,[2] as well as prominent criticism of the concept itself.[3] Although valuable and highly informative, in our analysis we take a broader, more structured and systematic approach, as mentioned earlier, by stepping away from specific projects and towards the underlying concepts.

5.3 ANALYSIS METHODOLOGY

We build our framework based on a combination of existing economic literature and careful consideration of possibilities and impossibilities in the cryptocurrency design space. For the former, we incorporate three classifications based on (1) monetary regimes [195], (2) exchange-rate arrangements studied by the International Monetary Fund (IMF) [110], and (3) stabilization techniques in employed in prac-

2 For example:
https://github.com/sdtsui/awesome-stablecoins
https://stablecoinindex.com/
https://media.consensys.net/the-stateof-stablecoins201879ccb9988e63
https://hackernoon.com/stablecoinsdesigninga-price-stablecryptocurrency-6bf24e2689e5

3 https://prestonbyrne.com/2018/03/22/stablecoinsaredoomedtofail/

tice by major central banks [24, 122]. For the latter, we systematically analyzed approaches proposed or used by various stablecoin projects and merged these insights into an abstract perspective. In that, stablecoins served a double-role of providing insights for the development of our taxonomy as well as being the subject of classification under said framework. Before diving into the details, we have to establish what classifies as a stablecoin in our analysis.

In [144], stablecoins are identified as cryptocurrencies "whose values are pegged to [i.e. stabilized relative to] some other fiat money or asset with inherent value". This definition of stablecoins, however, is exceedingly narrow. In traditional economics, stability can go beyond a long-term link to an foreign currency or some asset. As a consequence, we broaden the definition to cryptocurrencies "with mechanisms to mitigate fluctations in their purchasing power"[87]. The scope of our analysis and classification is limited to stablecoins that are (1) permissionless, (2) intended for general use as a currency, and (3) provide a white paper and website. By permissionless [272] we mean any coin or token that runs on a permissionless blockchain — specifically including IOU-Tokens such as Tether [193]. We exclude central bank digital currencies, pure utility tokens and stablecoins without a website or corresponding white paper.

[87] Purchasing power of a currency *describes how many units of certain goods, services or other currencies it can buy.*

Exploring, classifying and understanding the stablecoin landscape is a tedious task but a prerequisite for any further insights. Due to the short lived nature of many coins, any such perspective is necessarily a momentary snapshot of current projects, while our classification taxonomy goes beyond individual projects.

At the time of writing we identified 24 projects that fit these criteria[88]. Classification details for individual projects can be found in the Appendix, Section 7.2.1. Of these 24 projects, 13 are launched and traded on exchanges. As a preview of the empirical application of our taxonomy and subsequent discussions, we give an overview of their performance in Table 9. The table summarizes the mean, standard deviation, minimum and maximum price (in USD) of each launched coin according to data gathered from `https://coinmarketcap.com`. It can be observed that the projects show divergent performance and general statistic characteristics, due to differences in stabilization targets, guarantees and techniques. In the following section, we start with an investigation of stabilization techniques, i.e., the first overarching question of our framework.

[88] *Some projects issue stablecoins pegged to different national currencies. In these cases we exclusively address the coin pegged to the USD.*

5.4 STABILIZATION TECHNIQUES

Fundamentally, all stabilization techniques are based on the elementary economic model of supply and demand. The *price of a currency* can be modeled as the level at which its supply and demand meet each other on the market. A change in price is therefore due to changes in supply and/or demand — to maintain stability, any such change has

Projects	Obs.	Mean	Min.	Max.	Std. Dev.
NuBits (Nubits)	1587	0.819	0.030	1.264	0.332
BitShares (BitUSD)	1525	1.016	0.680	1.600	0.076
Tether (USDT)	1429	1.000	0.914	1.058	0.010
Karbo (Karbo)	904	0.299	0.005	2.066	0.418
Minex Coin (Minex Coin)	458	10.991	0.549	56.586	10.201
Maker (Dai)	404	1.002	0.939	1.053	0.010
Trusttoken (TrueUSD)	335	1.006	0.985	1.132	0.012
Digix (Digix Gold Token)	263	42.075	36.243	50.207	2.294
Sythetix (SUSD)	205	0.989	0.867	1.029	0.018
Stasis (EURS)	188	1.143	1.086	1.260	0.025
Centre (USD Coin)	118	1.013	0.983	1.037	0.008
Stronghold (USDS)	46	1.016	0.947	1.076	0.021
USC (USC)	32	0.912	0.656	1.027	0.152

Table 9: Basic descriptive statistics for available daily USD-prices of stablecoin projects until the 7th of February 2019.

to be counteracted. The price of a currency describes how many units of other currencies are given in exchange for it. The price can also be expressed in terms of other goods or services. Thus, in presented context the term can be seen as an equivalent to the currency's *exchange rate* and *purchasing power*.

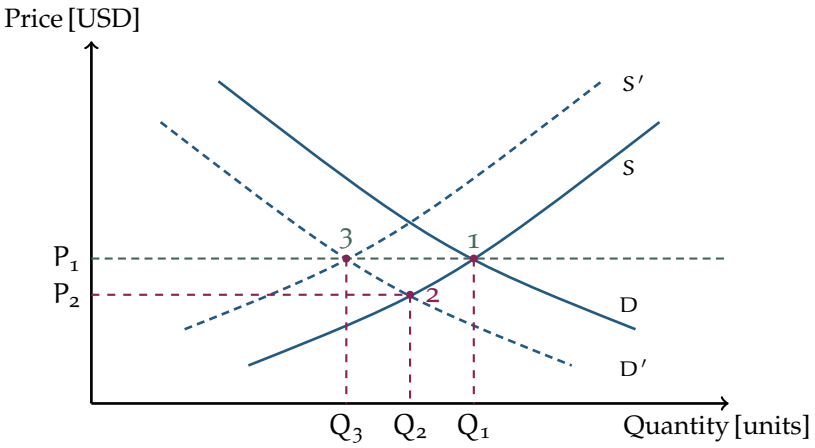


Figure 41: Examplary supply and demand model.

[89] *Thanks to Georg Gentzen for the TikZ-version of our initial sketch.*

Figure 41[89] illustrates this concept, with price (in USD) on the y-axis and quantity (coins in this case) on the x-axis. The solid S and D curves depict the money supply and demand, respectively. On a sidenote, the specific shape of the curves is merely an example. It abstracts the market where a cryptocurrency is exchanged for goods, services or other currencies. For some cryptocurrency setups, in the short run, the money supply is independent of the price, this makes

the money supply curve a vertical line that is shifted in the long run. The shape of the curve has no influence on the general rationale in the following explanations of stability techniques.

Both curves intersect at (Q_1, P_1), yielding an equilibrium quantity of Q_1 and price of P_1. Stablecoins aim to maintain a constant price, say P_1 in this example. Assume the demand decreases, i.e., users would purchase fewer coins at each price level, effectively shifting the demand curve to the left — from D to D'. The new equilibrium, the intersection of D' and S at (Q_2, P_2), has a smaller quantity (Q_2) and lower price (P_2) — which violates the aim of maintaining a constant price P_1. To recover, one can (i) increase demand (shift D' to the right), (ii) decrease supply (shift S to the left) or (iii) adjust both. Especially for cryptocurrency systems, demand is much harder to influence directly and instantaneously, therefore, supply is often the target of choice. This is depicted in Figure 41, where supply is adjusted, shifting S to S' which yields an equilibrium of (Q_3, P_1). Here, the quantity of coins on the market (Q_3) is smaller, but their price in terms of US dollars is back to the desired level (P_1).

Whatever deviation from the initial price, the stable purchasing power can, theoretically, be restored by adjusting supply and demand. Naturally, this model is a simplification and real-world examples are a lot more involved — however, it provides helpful insights for the analysis and classification of techniques for maintaining stability.

In the following, we give a systematic overview of techniques for influencing supply and demand and subsequently discuss potential risks and limitations. This overview compares techniques employed by traditional central banks alongside with techniques unique to cryptocurrencies. In our analysis we identified six major techniques: (i) collateralization, (ii) interest rates, (iii) currency interventions, (iv) open market operations, (v) dynamic block reward and (vi) dynamically burned transaction fee. The usage of those techniques is not mutually exclusive, a combination can be applied in practice.

5.4.1 *Tokenization of collateral*

Tokenization of collateral (or simply collateralization) links the coin supply to the demand, so that any change in the demand incentivizes market participants to change the supply accordingly. Each stablecoin token is backed by a certain amount of (crypto-)currencies, assets or fiat money. Users can create tokens by depositing an underlying backing, the so-called *collateral* and can redeem (destroy) tokens to receive their collateral. The entity which stores collateral might be a smart contract or centralized (as in the case of Tether [193]) — the limitations and drawbacks are discussed in Section 5.5.

The creation and destruction of coins through users provides a mechanism for supply adjustment. On the one side, when demand

increases, market participants can simply create new coins by depositing collateral, effectively increasing supply. Due to the excess demand, a coin might trade at a price higher than the value of the underlying collateral — in this case this arbitrage opportunity further incentivizes the creation of new coins. On the other side, when demand decreases, supply can decrease as well by redeeming coins in exchange for their collateral and therefore destroying them. Similarly, a coin might trade below the value of its collateral, creating arbitrage opportunities and therefore incentives to destroy coins.

Note that the described incentives (and therefore the success of this technique) rely on perfect transferability between coin and collateral: a coin can always be redeemed for its collateral and vice versa, without delays or any other friction. A violation of this assumption in practice might make this technique less efficient and therefore a coin subject to price swings.

A number of assets are conceivable as collateral. We distinguish three subcategory of collateralization: *direct*, *proxy* and *self-collateralization*.

In *direct collateralization*, each token is backed by the asset pegged to (i.e. the asset it is stabilized against). For example, if the goal is a stable exchange rate to the Euro, each token is backed by one Euro. This design resembles the approach of fiat currencies such as the Bulgarian Lev backed by Euro or Djiboutian Franc backed by US dollars. Examples of implemented projects include *Stably* [190] and *Tether* [193]. The already implemented concepts show relatively stable exchange rates to the USD. *Stably* [190] historically been able to maintain within a band of 10 % around the peg, and *Tether* [193] even within a band of 5 %. There are, however, examples with larger deviations.

In *proxy collateralization* each token is not backed by the targeted currency itself but instead by some other (crypto-)currency, asset or basket of assets[90]. Different from direct collateralization, there is a gap between collateral (e.g., Ether) and the stabilization target (e.g., USD): falling prices of the collateral may lead to insufficient backing.

[90] *Similar to what Facebook's Libra intended [218]*

Self-collateralization is a subform of proxy collateralization. In this technique, another token which is issued within the ecosystem of the cryptocurrency itself is used as collateral. The collateral risk is therefore elevated, since the fate of the ecosystem affects the stablecoin as well as its backing. An already implemented example is the stablecoin *BitUSD* [182] which is backed by the token *BitShares* [182]. While *BitUSD* [182] appeared relatively stable between 0.76 and 1.60 USD per *BitUSD* [182] for several years, it slumped below 0.67 USD in December 2018 when collateral prices declined.

For self and proxy collateralization, the gap between collateral and asset pegged to is mitigated with two (often combined) approaches: first, requiring more collateral than necessary (*over-collateralization*)

and second, enforcing automatic re-collateralization (*margin calls*). In over-collateralization, more backing is required than the actual price goal of the token would suggest. As an example, say a stable token backed with Bitcoin should trade at 1 USD, then over-collateralization would require to deposit Bitcoin worth 1.5 USD to create a token. This allows for some volatility of the collateral without risking that tokens become undercollateralized, i.e., when the backing is worth less than the price goal of the token.

Margin calls are triggered, if the value of the collateral falls below a predetermined value, the "margin", in order to avoid undercollateralized tokens. In a margin call either the creator of a token deposits more collateral or the collateral is offered for sale on the market in exchange for stable tokens. Given sufficient liquidity on markets, this effectively rolls back the creation of a token and decreases its supply.

5.4.2 *Use of interest rates*

Interest rates are an instrument to guide a decentralized adjustment of the money supply. For example, in the current real-world credit money system, most of the money is created when commercial banks issue loans to their clients [175]. The money stock decreases when loans are paid back or money in circulation is used to make deposits which lock money for a certain amount of time. Central banks set and adjust the base interest rate to influence interest rates of the commercial banks. The higher the rates, the smaller is the number of loans and the higher is the number of deposits in the system and the smaller the money supply becomes and vice versa for lower interest rates. The effectiveness of the technique ultimately depends on the decisions of the market participants to make *deposits* and to take *loans*.

Interest rates on deposits are in some stablecoin projects denominated as *parking* or *locking* fees. In this technique, users lock their coins in order to receive them back after a specific time with some additional reward (interest). The interest is paid by the system, most often through the minting of new coins. Higher interest rates make the currency more attractive for investors — demand increases. At the same time, as a higher fraction of currency is locked in deposits, supply decreases; at least temporarily. In the long run, supply only increases as deposits are paid back with interest rate. Among others, "Stableunit" [189], "Minex Coin" [187] and "Nubits" [188] employ interest rates on deposits.

Interest rates on loans are sometimes referred to as *stability fees*. Current implementations of loans in cryptocurrencies can be seen as a generalization of the collateralization technique: the stable token issued when depositing collateral is a loan on that collateral. However, to get the collateral back a user has to return the stablecoin and may also need to pay a non-zero interest. The interest rate is used to con-

trol the number of created coins. For instance, raising interest rates makes borrowing stablecoins more expensive — supply decreases. An example project that has launched a system with interest rates for both deposits and loans is *Maker* [185] with its token *Dai*. Since December 2017, Dai has deviated from a 5 % band around the 1 USD peg, with a single day at 0.94 USD.

5.4.3 *Currency interventions*

Currency interventions are a technique for a direct money supply adjustment. Here, an abstract monetary actor in the form of multiple persons and/or trading bots, intervenes in currency markets by buying and selling coins in exchange for the currency to which the stablecoin is pegged. When demand increases, coins are created and sold on the market for reserves. This increases the money supply to match the increased demand and subsequently normalize the price. In the opposite situation, when demand decreases, coins have to be bought back, decreasing supply and therefore stabilizing the price again. In contrast to collateralization where market participants are incentivized to stabilize the price through the backing with collateral, currency interventions require active intervention by some actor related to the stablecoin. Naturally, the purchase of coins requires that the monetary actor has currency reserves that can be spent on the market. Once the reserves are depleted, the exchange rate is governed by market forces, which can lead to a drastic change in the price and damage trust — further deteriorating the stability.

5.4.4 *Open market operations*

Open Market Operations (OMO) can be seen as a generalization of the currency interventions technique. A monetary actor manually or (semi-)automatically purchases external assets and pays them with newly minted money which increases the money supply. The system contracts supply by selling the assets back to the market[91]. For instance, if the Fed buys U.S. Treasury Securities on the open market, it effectively increases the supply of dollars. Selling these securities back to the market allows to decrease the supply again. A number of stablecoins implicitly or explicitly consider replicating this technique. The proposed designs, however, ignore certain safeguards often used by national central banks.

The most important of these safeguards are *eligibility* and *reversibility*. Eligibility demands that only highly secure and liquid third-party assets can built central bank reserves [23], [21], [22] or [25], [55]. This ensures that supply can be decreased in the future by selling the assets. The higher their price, the more money supply can be absorbed. Reversibility requires, that OMO is automatically reversed

[91] *We differentiate between currency interventions and OMO to highlight the specific feature of the former. Buying or selling the targeted currency against stablecoins implies a more effective impact on the mutual exchange rate as the supplies of the targeted currency and stablecoin move in the opposite direction simultaneously.*

after a predetermined period. This ensures that, by default, supply increases only short term.

To highlight negligence of the above safeguard principles, we differentiated between three sub-categories:

Standard OMO classifies OMO implementations satisfying the eligibility and reversibility safeguards. None of the proposed techniques in current projects can be classified as such.

Proxy OMO violates at least one of the safeguards. Proposals in projects like *Celo* [184] or *Augmint* [179] are examples.

Self-tokenizing OMO decreases supply not by selling external assets, but other assets generated within their own ecosystem. Examples are *Basecoin* [180], *Carbon* [183] and *Fragments* [186]. All these projects target a 1-to-1 relationship between their stablecoin and the USD. To decrease the money supply, the projects propose mechanisms that create special-purpose tokens that are sold for stablecoins which are then destroyed by the system. In theory, with such a design, supply can be decreased to any desired level. This is different from standard and proxy OMO that are restricted by the available reserves of external assets. In practice, many open questions remain (addressed in Section 5.5).

5.4.5 *Use of dynamic block rewards and dynamically burned transaction fees*

Instead of a pre-defined change in the money supply as in Bitcoin or Ethereum, the mining reward can depend on the current state of the system. If supply needs to be increased this can be done by increasing the mining reward. Since a very low or even negative block reward is not practical, this technique can only increase supply. Furthermore, increasing the mining reward is equivalent to "printing" money since currency is issued without any backing. Note that a variable block reward leads to variability in the hash rate due to varying incentives to increase/decrease mining power — elevating the risk of double-spending attacks [230]. To provide a way to decrease supply, some projects suggest *dynamically burned transaction fees*, i.e., a part of transactions fees is not given to miners but burned instead. Hence, the possibility to decrease supply is limited by the total volume of fees over a period of time.

5.5 STABILIZATION TECHNIQUES: DISCUSSION

In the preceding section we presented the stabilization concepts underlying the analyzed stablecoins. We purposefully disconnected the in-depth description of the techniques from the discussion of their merits and drawbacks for the sake of clarity, which is the center of this section.

5.5.1 *Tokenization of collateral*

While in theory collateralization in itself provides an elegant way to link money supply and demand through the action of market participants, it exhibits several risks and limitations that render this technique less reliable in practice. Direct collateralization, due the to link to fiat currencies or traditional assets, requires a trusted third party that manages funds, assets and the correct issuance of tokens. While the resulting counterparty risk can be remedied to some degree by, e.g., escrow accounts and diversified banking partners, the necessity of a trusted third party remains a major limitation.

Proxy collateralization could help to avoid above risks, since the collateral can be another cryptocurrency (e.g., Bitcoin or Ether). However, even if the counterparty risk can be eliminated, the requirement of a trusted price feed gives rise to the oracle problem (cf. Section 5.8). Furthermore, if the collateral's value fluctuates (as it is the case for cryptocurrencies), price risk of the collateral has to be mitigated. Margin calls are often cited as a remedy for collateral risk, however, margin calls require the assumption that markets for the collateral asset are liquid and large enough to allow for timely provision or absorption of collateral. This assumption might become an issue for young stablecoin projects or if expectations on the future development of the collateral are dire.

As a subform of proxy collateralization, self collateralization exhibits the same risks. Moreover, it suffers from additional systematic risk between the collateral and the stablecoin, as the value of the collateral is often a function of the future expected demand on the stablecoin.

5.5.2 *Interest rates on deposits and loans*

As discussed in Section 5.4, interest rates on deposits can reduce the money supply only temporarily and thus should be coupled with other techniques. When it comes to loans, an interesting question is whether under-collateralized loans can be implemented at all. This is the case in the regular economy, where wealth or future income can be used as collateral. We argue that this is impossible in permissionless cryptocurrencies due to the lack of strong identities and the resulting vulnerability to Sybil attacks [86]. If under-collateralized loans were implemented, rational actors would spawn multiple fake identities to obtain loans and free money.

Theorem 1. *In a permissionless setting without strong identities, under-collateralized loans enable arbitrage to the point where only fully collateralized loans are available.*

Proof. Let L be a loan that can be taken by depositing an amount of collateral C, with p_L and p_C denoting the respective prices. In an

under-collateralized setting $p_C < p_L$. A rational agent would seize the arbitrage opportunity, spend p_C on collateral and receive a loan with value $p_L = p_C + \epsilon$. The loan can be used to purchase more collateral and create more loans, generating a profit of ϵ_i in each step i, until the arbitrage opportunity closes due to increased collateral demand, i.e. $p_C \not< p_L$. Since there are no identities, the agent can refuse to repay the loans he has taken without any risk, locking the collateral forever and still generating a profit of $\sum \epsilon_i$. □

Even with smart contracts that enforce payments and interest rates, the lack of strong identities makes it easy to simply "exit-scam" the system, i.e., to generate a new debt-free identity and start over without negative consequences.

5.5.3 *Currency Interventions*

For currency interventions, the ability to maintain a peg during falling prices is limited by the amount of available reserves and the monetary authorities' commitment to make use of them. Once the reserves are depleted, the exchange rate is governed by market forces, which can lead to a drastic change in the exchange rate. The usage of currency interventions can, under certain assumptions, increase the vulnerability to speculative attacks (cf. Section 5.6.3). The interplay of full transparency of the system and gameability of the intended interventions is an interesting open question.

5.5.4 *Open market operations*

The negligence of safeguards by techniques classified as proxy OMO is no triviality. High quality (eligibility) of the assets seized by the cryptocurrency system prevents erosion of its reserves that can be used to buy back outstanding currency units. The programmatic reversal of open market arrangements ensures that a long term expansion of the money supply is not possible without manually overriding the default policy. While reviewed projects only allow using cryptocurrencies with a relatively long track record (Bitcoin, Ether), reversibility has not been proposed yet.

In self-tokenizing OMO not reserves but special-purpose tokens are sold against currency units. While the designs of the tokens vary, all provide some form of success-related monetary incentive that is payed out if a certain target price for the stablecoin is achieved. The incentive is paid out in form of newly minted money supply. The lower the probability of success, the higher the necessary incentives. Risk is either remunerated by higher relative ownership of future money growth or by a promised absolute increase of future minted money. Excessive use thus may either lead to reduced ownership in

risk remuneration or to an uncontrolled increase of promises of future remuneration in the money supply. Similar to currency interventions, OMO setups are vulnerable to speculative attacks (cf. Section 5.6.3). Lastly, the technique can decrease supply only in the short run, as long as token remuneration promises are not retracted.

5.5.5 *Dynamic mining reward*

Mining is a vital function of most cryptocurrency systems. The goal of making the money supply dynamic should be subordinated to the security and usability of the financial system. Low block rewards or high difficulty, e.g., in phases of stagnating demand for the currency, would lead to less incentive for miners to process transactions. This would necessarily lead to lower transaction throughput, which would not only reduce the liquidity of the coin, but also increase the risk of double-spending attacks. Moreover, as this technique cannot be used to reduce the money supply, it should be coupled with other instruments.

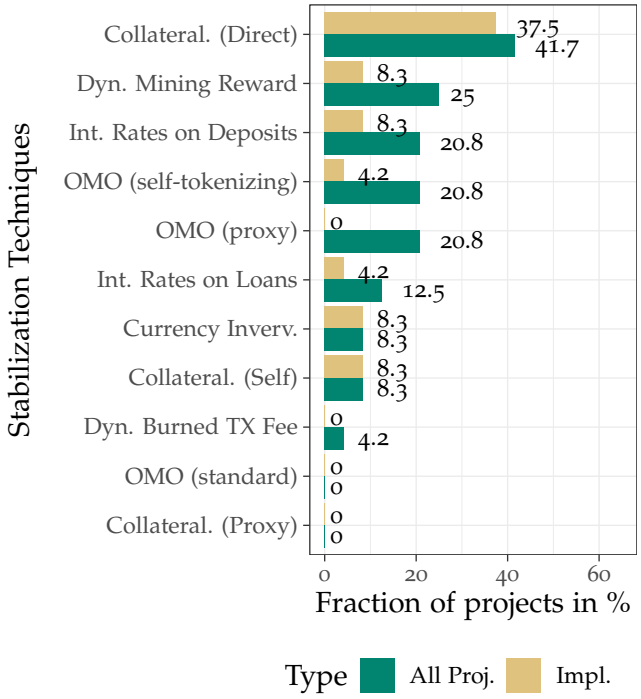


Figure 42: Planned and implemented stabilization techniques.

5.5.6 *Classification results and blank spots*

Figure 42 shows a full list of techniques discussed in this section as well as their prevalence of adoption in stablecoins projects, distinguished by planned and implemented.

The most popular technique according to our observations is direct collateralization. It is followed by the use of dynamic mining reward, interest rates on deposits and self-tokenizing OMO. None of these methods can permanently decrease money supply. Current stablecoin projects plan to launch primarily solutions which either require the participation of a trusted third-party or are focused on techniques that can decrease supply only temporarily and consequently are not sustainable in the long term[92].

[92] *Or assume constant growth in monetary demand.*

Interest rates on loans, currency interventions, standard OMO and maybe even proxy OMO might be useful techniques as they allow for decreasing the money supply permanently. However, exactly those have been worked on to a lower degree: although well over 40 % of projects plan some form of OMO, only around 4 % implemented their proposed setup. Note that established monetary policy standards find little acknowledgment — no project implemented the requirements of standard OMO, although other types of OMO are introduced. As there is little practical experience yet, risks and potentials of these techniques are hard to assess.

But also for less complex approaches there are blank spots. None of the reviewed projects has implemented dynamically burned transaction fees or proxy collateralization[93].

[93] *MakerDAO's Dai could arguably classified as proxy col. with Ether. However, we purposely classified Dai as interest-rate-based-stabilization as these are the main drivers of the incentive system specified in their white paper.*

5.6 EXCHANGE RATE REGIMES

So far we implicitly interpreted "stability" as stabilizing the price of each stablecoin to *exactly* 1 EUR or 1 USD. While this seems to be an intuitive approach, other so called *exchange rate regimes* are possible.

5.6.1 *Types of exchange rate regimes*

We base our framework upon a taxonomy of the IMF [110] which splits exchange rate regimes into three main types: *hard pegs*, *soft pegs* and *floating regimes*. These regimes are ordered hierarchically with sub-types in our framework, (cf. the Appendix, Section 7.2.2).

A *hard peg* can come in one of two flavors: arrangements without legal tender and so-called *currency boards*. In an arrangement without legal tender a country chooses to simply use a well-known foreign currency like the USD instead of issuing their own.[94] We neglect this case since it is clearly not useful for cryptocurrencies: it would essentially mean to avoid them altogether. In a currency board the domestic currency is backed 1:1 (or more) by reserves of the foreign currency [107]. That is, for every issued unit of domestic currency, there has to be at least one unit of the foreign currency in the reserves.

[94] *Note that, on a more general level, currencies can also peg against external assets (e.g., gold).*

Different from hard pegs, *soft pegs* are characterized by weaker commitments to a fixed rate, i.e., there does not need to be a 1:1 backing with reserves. Soft pegs come in a variety of different flavors.

The most important are: conventional pegs, pegs with horizontal bands and crawling pegs.

A *conventional peg* is defined by the level of allowed deviations. The IMF specifies a maximum fluctuation of 1 % over a time period of at least six months around the pegged value. A weaker form is the so-called *peg with horizontal bands*, where the exchange rate is allowed to fluctuate within a pre-announced (wider) range around the pegged value. These peg types share a common property: the exchange rate is constant over time. In contrast, *crawling pegs* allow for a gradual adjustment in the exchange rate.

Last but not least, if the exchange rate is *floating*, little to no guarantees are given about the stability of the value. Instead, the exchange rate is determined by market forces to a large degree and monetary interventions are kept to a minimum. Due to the fact that free floating can lead to high volatility, some countries intervene aggressively against short term fluctuations (cf. Section 5.4). This practice is known as *smoothing* or *floating with interventions*.

5.6.2 *Classification results and blank spots*

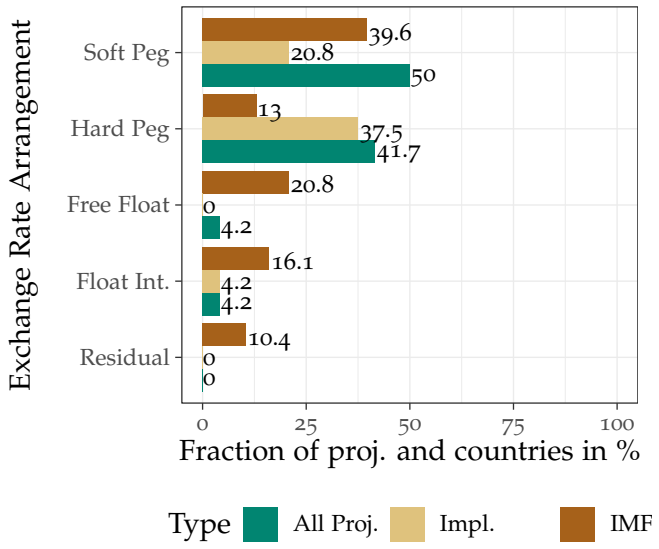


Figure 43: Exchange rate arrangements: stablecoins and national currencies.

Figure 43 shows a comparison of exchange rate arrangements in stablecoins and traditional central banks.[95] The data for central banks stems from a study of the IMF in 2016 [110].

The majority of stablecoins, more than 90 %, commit to achieve some kind of peg. As in traditional central banking, in cryptocurrencies one has to distinguish between what is announced (de-jure) and the historical exchange rate (de-facto). De-jure the majority of

[95] *Note that the category "residual" refers mainly to countries with frequently changing monetary policy approaches.*

Projects	±1 %	±5 %	±10 %	±20 %
Tether (USDT)	12.11	1.05	0	0
Maker (Dai)	25.25	0.50	0	0
Trusttoken (TrueUSD)	30.75	0.60	0.60	0
Sythetix (SUSD)	33.66	3.90	0.98	0
Centre (USD Coin)	66.10	0	0	0
Stronghold (USDS)	80.43	8.70	0	0
BitShares (BitUSD)	70.95	30.23	13.84	4.26
NuBits (Nubits)	42.22	27.22	26.59	24.07
USC (USC)	56.25	31.25	31.25	25

Table 10: Percentage of days for which certain bands around the 1 USD peg are violated.

stablecoins commit themselves to a fixed 1:1 correspondence to the USD in their white papers. Almost half of all projects, tries to enforce this by establishing a currency board and storing the fiat currency pegged to. Implemented examples include *Tether* [193], *Stasis* [191] and *Trusttoken* [194].

The remainder (50 % of all projects), does not implement a currency board and are therefore classified as a soft peg. Examples include *Maker* [185], *Stasis* [190], *Nubits* [188], *Synthetix* [192] and *Bitshares* [182]. Most abstain from explicitly specifying bands and are therefore conventional soft pegs[96].

[96] *Some projects mention "some" corridor around the peg. As bands are supposed to be predetermined and announced for accountability reasons, we still classify them as conventional pegs.*

Table 10 shows the fraction of daily closing prices violating certain thresholds between the launch of the respective coin and February 7, 2019. The table contains the subset of coins that pursue a 1-to-1 peg to the USD. De-facto, none of the already launched cryptocurrencies meets the demands that would be posed by the IMF for a working conventional peg. Interestingly, even *Tether* [193] and *Trusttoken* [194] violate the requirements, despite implementing currency boards. These fluctuations may stem from uncertainty caused by a perceived lack of transparency and accountability or lower market liquidity. National currencies in turn tend to use floating arrangements more often. Although the majority of analyzed stablecoins pursues pegs, concepts for floating arrangements with interventions are also in development, e.g., *MinexCoin* [187] proposes interventions to keep daily price changes from exceeding 5 %.

5.6.3 *Vulnerabilities to speculative attacks*

The usefulness of soft pegs is disputed in economic literature. This standpoint is called the *bipolar view* [102, 269], and is broadly supported by mainstream economists [63, 91, 92, 102]. The bipolar view suggests that there are only two long-term viable options for currency regimes

that care for exchange rates: hard pegs or floating with interventions. Arguments include the short life expectancy of soft pegs and their vulnerability to speculative attacks [102].

Speculative attacks on soft pegs are known from traditional central banking [151, 211], but the threat is equally applicable to cryptocurrencies. This is especially relevant considering that 50 % of stablecoin projects plan on using soft pegs.

If the market believes that a fixed exchange rate is not sustainable, investors will start speculating against it to make a profit in the event that it eventually breaks. To counteract, central banks have to invest resources to defend the peg, which is costly and oftentimes unsuccessful [212]. A vivid illustration of unsuccessful peg defense is the Bank of England's attempt to maintain a fixed Great Britain Pound (GBP)-European Currency Unit (ECU) exchange rate during a speculative attack in 1992 lead by the hedge fund "Quantum"[97].

97 ECU was an artificial currency used within the European Monetary System before the introduction of the Euro in 1999 [90].

The investors and, subsequently, other market participants followed a simple algorithm:

1. Borrow GPB and sell them, at market price, for German Marks (DM); this is called a *short sale*.

2. When the peg fails and the exchange rate drops, buy back GBP at a cheaper price and return to lender.

The selling of borrowed GBP for DM increases the supply of GBP and reduces the supply of DM. Due to the fundamental principles of demand and supply, this in turn leads to an appreciation of DM and depreciation of GBP. To counteract and maintain the target exchange rate, the Bank of England bought the excess GBP on the foreign exchange market in exchange for their DM reserves. Furthermore, the Bank of England also increased the base interest rate. Buying the excess supply of GBP aimed at reducing the supply of GBP on the markets, whereas the increase of the interest rate aimed to increase the demand for GBP. However, after spending 15 billion USD in foreign reserves in only a single day, the Bank of England eventually had to abandon the pegged arrangement [90, 130]. The exchange rate on the market dropped, yielding an estimated profit of 1.5 billion USD[4].

There are two main sources that make speculative attacks on pegs highly probable [151, 211]: unsustainably constructed pegs and untrustworthy commitment to defend the peg. A peg is unsustainable if the central bank lacks sufficient reserves to invest in the case of a speculative attack. In cryptocurrencies this vulnerability is increased further, since they often have a small market capitalization and little reserves in comparison to traditional financial assets and currencies. Furthermore, the complete transparency of reserves due to the trans-

4 https://www.forbes.com/sites/steveschaefer/2015/07/07/forbes-flashback-george–soros-british-pound-euro-ecb/4e0e93346131

parency of the blockchain makes it easy for speculative attackers to validate the success of their strategy [37].

Adapting [151] to cryptocurrencies, consider a situation where the natural floating exchange rate would be lower then the peg. Potential reasons for this mismatch might be new vulnerabilities or general uncertainty in cryptocurrencies due to regulation. In both cases, the stablecoin system would need to intervene over longer periods of time, draining its reserves. Two long-term outcomes are possible: (1) the peg holds or (2) the currency finally depreciates when the intervention capabilities are depleted.

Now consider a user of the coin who chooses to *hold* her position. This user will have no payoff in case (1) and negative payoff in case (2). Therefore, the expected payoff from holding is negative. In contrast, if the user *sells* her coins, the sale can be reverted with little cost in case (1) and can avoid a loss in case (2). Therefore, the payoff for selling is higher than for holding. Rational market participants will sell their holdings.

The expected payoff of the sell strategy can even be increased through leverage by borrowing coins[98]. Speculators might borrow large quantities of stablecoins at the pegged price and sell them on the exchanges: if the stability system succeeds in defending the peg, speculators can buy back the coins at the peg and revert their positions with little losses. If the attack depletes the reserves of the system, the peg can no longer withstand the selling pressure and the exchange rate depreciates and becomes floating. Attackers can now buy back the stablecoins much cheaper, give back borrowed coins and keep the difference as profit.

[98] *Since efficient credit markets have not yet developed for all cryptocurrencies, the transaction costs to execute speculative attacks might be increased.*

5.6.4 *Peg hard or do not peg at all?*

As discussed, the bipolar view suggests hard pegs, float or float with interventions.

While conclusions transferred from monetary policy studies should be treated with caution, the bipolar view still offers insights useful for cryptocurrency systems: hard pegs using full direct collateralization and floating exchange rate arrangements are less vulnerable to speculative attacks then soft pegs. This explicitly holds for all soft peg implementations that do not allow for the retraction of most of the money stock in any kind of market situation.

As discussed, currency interventions and open market operations (OMO) that contract money supply by selling limited reserves are definitely concerned. Self-tokenizing OMO and interest rates on deposits buy back coins against self-issued securities with potentially unlimited supply. As discussed in Section 5.5, buyers of these special-purpose tokens are incentivized by a share in newly minted money in the case of *long-term* increasing demand for the stablecoin. As discussed in

Section 5.6.3 speculative attacks entail almost no risk or costs for the attacker, making repeated attempts in the *short run* possible. While in the presence of speculators for and against the peg the first series of speculative attacks might be neutralized, claims for risk remuneration will stack up quickly. Leading to a decrease in relative ownership of future remuneration, this will decrease the demand for the used special-purpose tokens with every round of attack. Missing demand for the self-issued tokens makes it impossible to absorb money supply and defend against the attack. Further research is strongly encouraged as the above setup is quite popular: almost 40 % of analyzed projects consider it.

We do question though, if all kinds of soft pegs are equally vulnerable in the case of cryptocurrencies. Soft-pegs relying solely on full proxy and self-collateralization promise to provide sufficient collateral to buy back the complete stock of money at any moment of time. This, in turn, makes them immune to the above described attack [212] — as long as the collateral remains sufficiently stable.

5.7 MONETARY REGIMES

Up to this point, we used the notion of "stability" in the sense of low exchange rate volatility. In the following, we zoom out further, stressing the difference between

- stabilizing the *amount of another currency* one cryptocurrency unit can buy (exchange rate) and
- stabilizing the *amount of goods and services* one cryptocurrency unit can buy (purchasing power).

Stable purchasing power is a goal which traditional central banks and stablecoins both pursue. Stability of prices can be measured, e.g., through a basket of goods in a consumer price index (CPI). In practice it can be influenced only via indirect measures. These encompass interest rates, exchange rates and many others. The respective choice of tools constitutes the monetary regime. Each monetary regime chooses a certain core variable, the so-called *nominal anchor*, to construct its monetary policy around. The chosen nominal anchor is used to choose practical applications of monetary instruments and to evaluate their effectiveness. It can be seen as the central element of the monetary regime and as the measurement variable around which central bank communication and also accountability line up.

Thus, while stable purchasing power is the overarching goal — fixing exchange rates (so called *exchange rate targeting*) is only one of several strategies to achieve it. Other monetary regimes focus on other factors than the exchange rate, namely *monetary targeting* and *inflation targeting*.

Monetary targeting uses the amount of money as its nominal anchor [196]. Assuming predictable velocity of money, the so-called Quantity Equation of Money can be used to calculate the necessary money supply to achieve a certain level of prices [109][99]. Correspondingly, adjusting the money supply is a key means of intervention for a central bank in such regimes.

Inflation targeting uses the change in a consumer price index as nominal anchor [35]. The most characteristic differences to monetary targeting lies in the publication of numerical inflation targets and the commitment to hit them. Additionally, also commitment and ability to achieve the inflation target, emphasis on transparency and increased accountability are quoted characteristic of inflation targeting [196].

[99] *Different versions of the quantity equation of money arose after being popularized by [103]. All have in common that they relate the aggregated flows of money to aggregated flows of goods and services. The equations offer different perspectives on the demand of money.*

5.7.1 *Regime-inherent aspects*

While *exchange rate targeting* is a popular arrangement for cryptocurrencies and countries alike, it exhibits major drawbacks. First, as stated in [212], exchange rate targeters lose the ability to pursue independent monetary policy. Moreover, inflationary tendencies and shocks are imported directly into the cryptocurrency. Third, as discussed in Section 5.6.4, exchange rate targeting can lead to vulnerabilities to speculative attacks. On the other hand, exchange rate targeting offers convincing advantages from the perspective of cryptocurrencies. First, pegging the value of a cryptocurrency to some other currency or asset can reduce the volatility drastically, since price fluctuations of, e.g., USD, are magnitudes smaller than in most cryptocurrencies [276]. Second, not even the most mature cryptocurrencies do succeed to be used as unit of account for the purchase of goods or services [120, 276], so that prices are not typically quoted in cryptocurrency units. Therefore, also from a usability perspective it is a reasonable choice to strive for a stable relationship to fiat currencies.

Inflation targeting poses the obvious challenges of the definition and tracking of an adequate basket of goods and services. More importantly, goods usually are denominated in some national currency. Purchasing power fluctuations of a volatile cryptocurrency measured by a basket of dollar denominated goods should mainly be caused by exchange rate variability. As a consequence inflation targeting and exchange rate targeting would be close to equivalent.

Monetary targeting in a sense is already implemented by traditional cryptocurrencies with predetermined block reward. More sophisticated monetary targeting approaches might use literature around rule based monetary policy (e.g. [173], [174], [158] or [172]) as first starting point. If additional measures against exchange rates are to be taken, managed floating regimes as promoted by [121, 156], might offer a simple but sustainable alternative to exchange rate targeting. This approach, however, mitigates only short term fluctuations.

5.7.2 *Classification results and blank spots*

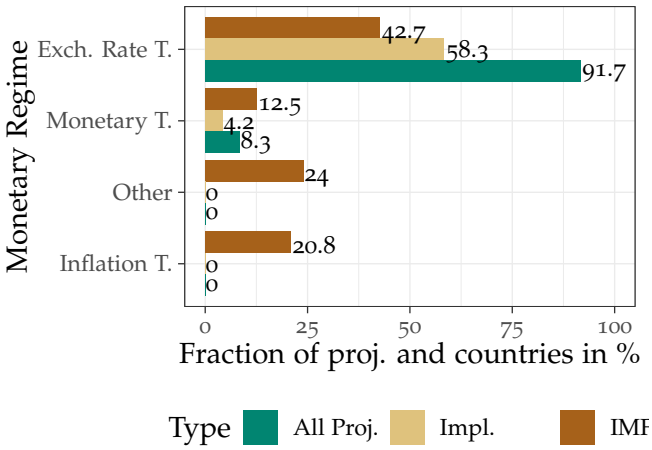


Figure 44: Monetary regimes: reviewed projects and central banks.

Applying the conceptual insights of our taxonomy to real-world projects, we see in Figure 44 that the prevalence of exchange rate targeting in the reviewed projects is in stark contrast to traditional central banks (numbers for countries from [110]). Exchange rate targeting, accounting for most than 90 % of all projects, is the clear favorite of current approaches to cryptocurrency stabilization. Only 40 % of countries use a certain exchange rate as their currency's nominal anchor. While monetary targeting in combination with short term exchange rate smoothing could be an interesting alternative, it has largely been ignored by cryptocurrencies.

5.8 DECENTRALIZATION AND TRUST

Last but not least, we depart from mainly economic questions and will now discuss the design of stable cryptocurrencies in terms of their "decentralization" and "trustlessness" — notions that cryptocurrencies are commonly associated with and potentially owe their popularity to. Ideally, stablecoins strive to achieve value stability in a permissionless setting, similar to those of Bitcoin or Ethereum. As we have seen throughout this thesis, the implication of weak identities that a fully permissionless setting entails [272] opens up potential attack vectors through the Sybil problem (cf. Section 2.4.1). The lack of a strong cohesion between identity and entity renders various building blocks for stablecoins impossible — therefore, with the current state of knowledge, it is an open question whether an effectively price-stable cryptocurrency can at all be realized in a fully permissionless setting.

For example, techniques based on the collateralization or holding of "off-chain" assets (such as classical currencies) are *inherently incom-*

patible with a fully permissionless setup. Neither the actual existence nor the correct management of off-chain collateral can, in general, be secured through purely technical means. Rather than that, a form of trust, either in one ore more well-known custodians or in a surrounding legal framework and its enforcement mechanics, must be assumed.

Techniques based on collateralization, interest rates and OMOs are in principle compatible with a fully permissionless mode of operation as long as they act on assets whose ownership can be securely tracked and managed in a permissionless manner (e.g., are recorded on the same permissionless ledger). Even then, however, a fully permissionless mode of operation is only possible under a significant caveat — the existence of a secure (permissionless) oracle for the chosen nominal anchor. Oracles are system components that transfer "external" information onto the blockchain. Monetary information, such as the price of the cryptocurrency relative to another currency, are required for monetary policy mechanisms. They are typically not natively generated "on-chain" and must therefore be provided by an oracle. Oracles can be trivially realized using a trusted party that vouches for the correctness of data by means of cryptographic signatures. However, this clearly reinstates a globally trusted actor (or a group thereof). Completely permissionless oracles are, on the other hand, still an active research field, with no sufficiently secure solutions for, e.g., transferring price data, currently in sight [10]. It is possible that secure permissionless oracles for arbitrary data are a theoretical impossibility due to the Sybil problem (cf. Section 2.4.1) and game theoretic weaknesses [145, 268]. In such a case, a final possibility for the realization of completely permissionless stablecoins remains in the deepened investigation of on-chain *proxy variables* for relevant nominal anchors like current prices. We are currently aware only of the current mining hash rate, as materialized, e.g., in the timing between blocks and the current mining difficulty, as a potentially viable representative of this class. More research is needed here to further test the viability of this approach, especially in respect to incentive-compatibility, gameability and security (cf. Section 5.5.5).

5.9 CHAPTER SUMMARY

5.9.1 *Summary*

In this chapter, we systematically explored the enigma of monetary stabilization in cryptocurrencies. To this end, we combined insights from economic literature as well as blockchain-specific properties into a comprehensive classification taxonomy. Equipped with this framework, we classified 24 stablecoin projects according to their stabilization mechanism, exchange rate agreement and monetary regime.

This approach enabled us to go beyond individual proposals and projects and instead put the focus on overarching concepts and inherent limitations.

We find that direct collateralization, i.e., backing a crypto token directly by depositing a fiat currency, is the most prominent stabilization technique. Moreover, our findings show that almost 38 % of surveyed coins promote a problematic combination of exchange rate targeting and either limited reserves or a potentially unlimited supply of self-issued tokens to reduce the coin supply. There are strong indicators that the above setup can result in a vulnerability to speculative attacks. Zooming out, we suggest that short term smoothing of exchange rates might offer a sustainable alternative to exchange rate targeting — the current focus of over 90 % of projects.

We identified a number of further opportunities for technical and economic research on cryptocurrency stabilization, such as on the resilience of self-tokenizing techniques, on the viability of secure permissionless price feeds for informing policing decisions, and on the actual effectiveness of monetary policy given the complete transparency of both the policy and its enforcement.

5.9.2 *Conclusion*

A central conclusion from our work presented in this chapter is that robustness of systems aiming for monetary stability can not be assessed by technical means alone but instead requires a more holistic perspective — which has been adopted by other researchers as well [126, 145, 146, 222, 268]. As an example, consider direct collateralization, the technique of choice for many stablecoin designs with prominent examples like Tether [123]. Although economically one of the more sustainable approaches, through the interface with fiat currencies there is always an inherent aspect of centralization. In the case of Tether this would be the foundation/bank that manages the deposited funds, yielding counterparty risk that permissionless settings strive to avoid.

Other employed techniques more suitable for permissionless systems that are technically sound suffer from their inability to contract the money supply and are therefore economically unsustainable in the long run. In subsequent works, fellow researchers have showed that (speculative) attacks as well as adverse market conditions can lead to deleveraging spirals, effectively breaching the goal of monetary stability [268]. Factors for these spirals and attacks include, e.g., price swings in collateral (in the case of proxy-collateralization) and subsequent margin calls of deposits, re-organization of transactions and blocks by miners, as well as price (feed) manipulation. Especially the second point, re-organizations of the order of transactions through bribes to miners highlights a crucial issue: the application layer (the stablecoin in this case) can adversely affect the infrastructure layer

by incentivizing miners to create and/or switch to specific forks. Reversely, miners on the infrastructural layer may leverage their power of determining the order of transactions through, e.g., exploiting arbitrage opportunities and front/back-running [145]. These multitude of potential attack vectors makes mitigating potentials risks and stabilizing coins even under adversarial conditions challenging and subject to future research [145, 146].

6

CONCLUSION

In this thesis we investigated whether the widespread narrative of blockchain application's inherent decentralization entailing robustness against DoS attacks and censorship, is appropriate. To this end, we identified three main building blocks or layers of blockchain applications in practice: infrastructure, storage and monetary stabilization. Each layer was studied separately by investigating representative systems and their potential interconnections. By combining empirical analyses and theoretical considerations we were able to provide new insights and a fundamental understanding of these systems and the layers that they represent. Additionally, this allowed us to reason about the decentralization and robustness of blockchain applications based on top of these systems. Regarding the infrastructure and storage layer, we adopted a technical, network layer-based perspective on two widely used P2P systems: Ethereum and IPFS. When investigating monetary stabilization, we complemented the technical side with an economic point of view.

In order to adopt a network layer perspective, in Chapter 2 we first introduced the concept of P2P networking, Bitcoin, Ethereum and attack vectors that decentralized systems face, in particular, Sybil and eclipse attacks. While the Sybil attack is, potentially fundamentally unsolvable in permissionless settings, we have seen viable design choices introduced in Bitcoin to raise the necessary resources for launching successful Sybil or eclipse attacks to impractical degrees.

In Chapter 3, we saw that Ethereum's overlay does not adhere to these design choices and is therefore vulnerable to network layer attacks. We demonstrated this vulnerability by presenting a low-resource Eclipse attack which specifically targets the Kademlia-based peer selection logic. In particular, by inserting a small number of carefully crafted Sybil identities which are favored over other, benign, nodes during connection establishment, we were able to successfully eclipse a peer with only *two* IP addresses from distinct /24 networks. As Ethereum is widely used as infrastructural basis for blockchain applications, e.g., in the context of DeFi and Aragon DAO, eclipse attacks can be leveraged for double spending funds or general DoS of applications. Although eclipse attacks are rarely used for double spends in practice due to other attack vectors with better risk/reward profiles, DoS attacks against groups of nodes are still a genuine threat, especially given the little amount of resources required to do so. Lastly, we reviewed the countermeasures introduced to Ethereum in the process of responsible disclosure with the Ethereum foundation.

While the immediate threat is mitigated, we concluded that Kademlia is not a suitable approach for building a robust overlay.

Kademlia also serves as the basis for IPFS, the representative system on the storage layer studied in Chapter 4 — similar attack vectors as in Chapter 3 were therefore conceivable. However, IPFS uses a hybrid overlay structure between Kademlia-based DHT lookups as well as one-hop flooding of content requests for finding data. Whereas this unusual combinations renders eclipse attacks significantly more difficult, it also enables extensive monitoring possibilities which we exploit to paint a holistic and comprehensive perspective of IPFS' network and content layer. To this end, we crawled the Kademlia DHT and monitored data requests. Based on these extensive measurements, we concluded that through sacrificing performance for robustness, IPFS achieves a high resilience against censorship and DoS attacks, while only showing moderate degrees of centralization.

Lastly, we turned our attention to monetary stabilization in cryptocurrencies in the form of stablecoins in Chapter 5. These projects try to achieve the best of two worlds: a permissionless cryptocurrency combined with the value stability of traditional fiat currencies. For gaining a thorough understanding and to asses their stability, we adopted a hybrid perspective between economics and computer science, since these analyses are not only bound to technical considerations, but also related to stability under adverse market conditions and rational agents. Combining insights from monetary theory with specificities of cryptocurrencies, we presented a comprehensive taxonomy on monetary stabilization in blockchain-based currencies. The application of this taxonomy yielded a classification and deep understanding of the stablecoin landscape, which allowed us to go beyond individual projects and instead reason about fundamental properties and limitations. We concluded that many stablecoin designs are either not fully permissionless, due to the involvement of trusted parties, or use problematic combinations of stabilization techniques and stability goals which could render them vulnerable against economic attacks.

In summary, each chapter contributed new insights into the studied systems that, particularly in the case of IPFS, go well beyond blockchain applications. Based on a thorough empirical understanding, we were able to embed the analysis of each system and the respective insights into the broader context of robustness in decentralized applications. Furthermore, we bridged the gap between scientific research and practice through transfer of knowledge. In particular, we collaborated with the Ethereum Foundation and ProtocolLabs (the creators of IPFS and related libraries) — providing insights from our research but also strengthening our understanding through fruitful discussions and different perspectives.

Towards future works, the interdependencies between different layers, i.e., infrastructural, storage and monetary stabilization, need

further investigation. Their complex interplay bears potentials for security risks and attack vectors that have yet to be studied. This thesis provides an important step into that direction by diving into each component separately and assessing each component's impact on potential applications.

7 APPENDIX

7.1 IPFS

7.1.1 *Complete Crawl Report Feb. 2021*

7.1.1.1 *Basic Statistics*

This is the complete report on the IPFS crawl from 2021-01-24 until 2021-02-07, i.e., C_2. In the 14 d of crawling we performed a total of 5039 crawls in total, each of which took 3.8 min to complete, on average (the runtime distribution of crawls is depicted in Figure 30 in Section 4.7). We found an average number of $1.1 \cdot 10^4$ nodes per crawl; the crawler was able to connect to 47 % of them. In total, we found $1.37 \cdot 10^5$ distinct nodes during the whole crawl period. The

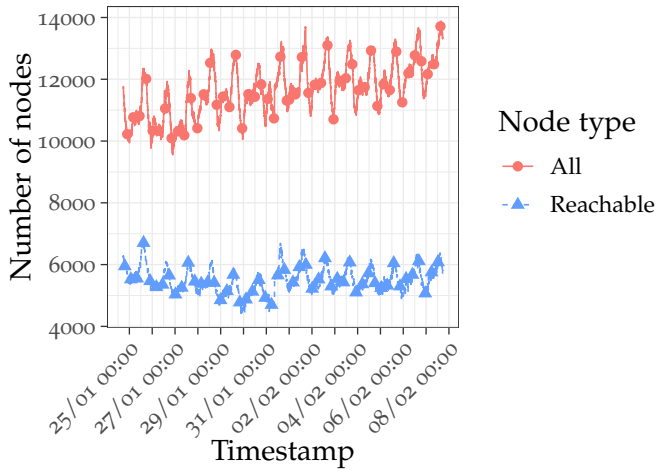


Figure 45: Number of nodes over time, distinguished by all and reachable (=answered to our query) nodes. Times are in UTC.

number of nodes over time is depicted in Figure 45, where time is on the x-axis and the node count on the y-axis. The figure distinguishes between all nodes and nodes that were reachable, i.e., the crawler was able to establish a connection to these nodes. In comparison to C_1, the number of total nodes has decreased significantly, as these nodes are now treated as clients and do not participate in the DHT. Nevertheless, a periodic pattern in the number of nodes is still visible; a Fourier analysis yields a dominant frequency of 24 h. The fit to the data is not as close as in C_1, as can be observed in Figure 46, due to the upwards trend in the number of nodes and other phase-shifted frequencies.

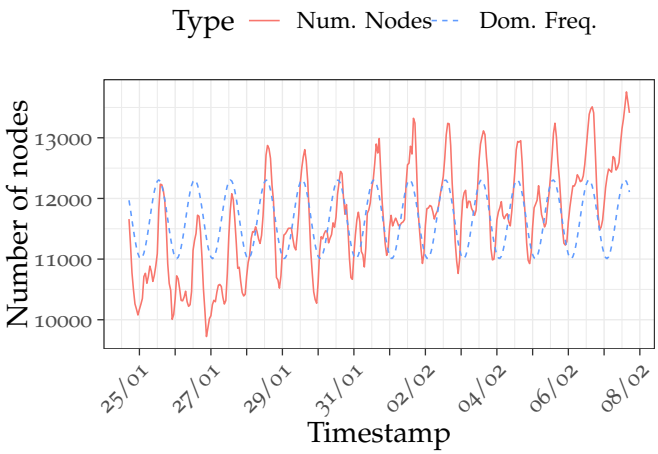


Figure 46: Fourier analysis and dominant frequency of the number of nodes. Ticks on the x-axis correspond to 00:00 UTC.

Session Duration	Percentage	Number of sessions
5 minutes	75.72	6.72e5
10 minutes	62.84	5.57e5
30 minutes	40.62	3.60e5
1 hour	25.26	2.24e5
1 day	1.0	8e3
6 days	0.16	1e3

Table 11: Inverse cumulative session lengths: each row gives the number of sessions (and total percentage) that were *longer* than the given duration.

Using the crawl data, we interpolate the uptime of nodes, also referred to as a session, the results of which are gathered in Table 11. To this end, we use a simple assumption: if we see a node in consecutive crawls, we assume that it was online in the meantime. Therefore, the crawl duration (3.8 min) is the granularity with which we can measure the length of a session. The table depicts the inverse cumulative distribution of uptimes: each row shows the number of sessions (and total percentage) that were *longer* than the given duration. In comparison to C_1, session lengths have increased significantly: in Nov. 2019, only around 2 % of all observed sessions were longer than 30 min, whereas in Feb. 2021 roughly 40 % of all sessions exceeded this duration. Similarly, the share of sessions longer than one hour is sixty times larger in C_2 than in C_1.

With newer versions of the crawler we also gather agent versions, the results of which are depicted in Figure 47. The plot shows the number of nodes per agent version, restricted to the ten most-seen versions throughout the entirety of the crawl. As mentioned in Section 4.7, a large fraction of DHT-enabled nodes sends the "storm"

agent version string. These nodes are part of the storm botnet that infects, e.g., windows machines and consumer hardware [224]. Storm uses IPFS as a command & control infrastructure. Interestingly, the daily fluctuations in the number of storm agents might serve as an indicator that these nodes are in fact operated by private individuals whose computing device is infected with storm. This daily pattern of storm nodes is likely contributing one "signal" that is superimposed with other periodic patterns in Figure 46.

One might object that the influence of storm nodes, e.g., through such a superposition, should be differentiated in the analysis or even excluded altogether in order to focus on legitimate uses of IPFS. We would argue that a holistic view of IPFS includes all usages, regardless of their morality: it is what IPFS is partially used for — comparable to TOR [256]. "Legitimate usage" implies a normative boundary and leaving out the storm botnet from analyses does not necessarily yield exclusively legitimate usages, as became evident in our discovery of youth pornography on IPFS which we reported to German law-enforcement authorities. Furthermore, extracting storm nodes is only possible to some extend, due to their verbose agent version string. It remains unclear if other, more normally appearing nodes, belong to the same or another botnet, hence, even at the current moment, an exclusion of respective agent versions alone is potentially insufficient to eliminate all relevant storm nodes in the analysis. Additionally, it is conceivable that the botnet will eventually switch to a more sophisticated and stealthier method of identification, making it hard to keep track of these nodes in that scenario.

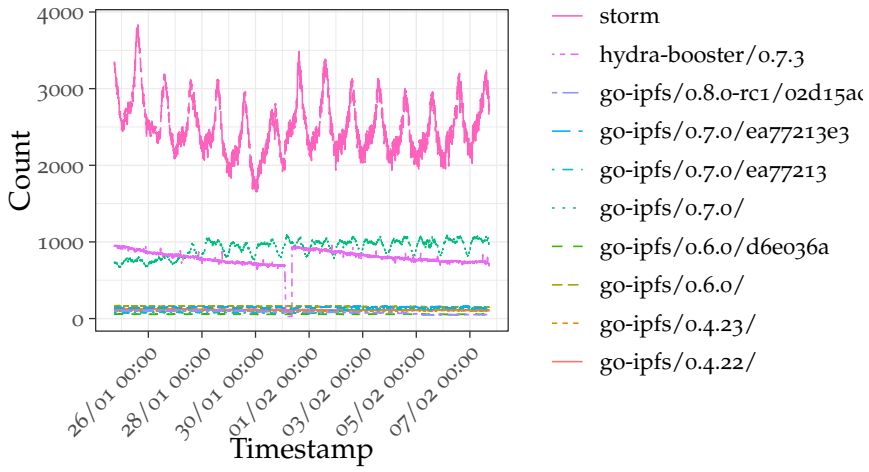


Figure 47: Agent version distribution over time. Depicted are the ten most-seen versions utilized by 88.4 % of all nodes.

All			Reachable		
Country	Count	Conf. Int.	Country	Count	Conf. Int.
CN	3700	±20.1	US	1380	±3.6
US	2290	±4.3	CN	680	±4.1
KR	796	±2.1	KR	640	±1.8
Unknown	772	±1.7	HK	420	±4.2
HK	650	±6.0	Unknown	317	±1.8
DE	414	±0.3	DE	289	±0.1
LocalIP	352	±1.2	TW	185	±2.2
TW	290	±2.9	UA	160	±1.5
FR	221	±0.5	FR	155	±0.3
UA	216	±1.9	RU	148	±0.9

Table 12: The top ten countries per crawl, differentiated by all discovered nodes and nodes that were reachable. Depicted is the average count per country per crawl as well as confidence intervals.

7.1.1.2 *Node Distribution over Countries and Protocol Usage*

Table 12 depicts the top ten countries, distinguished by all discovered nodes and for nodes that were reachable by the crawler. These ten countries contain 83.5 % (79.8 % in the case of reachable nodes) of the whole network.

The table shows the mean count per country per crawl as well as 95 % confidence intervals (assuming a student t distribution of observations, which is a reasonable assumption when considering means). Furthermore, the statistics are distinguished between all and only reachable peers. This means a row should be read as follows: for country c we have seen "Count" peers from that country per crawl, on average. The "Conf. Int." column shows how large the spread around this average value is.

Of all seen nodes 3.02 % only provide local or private IP addresses, thus making it impossible to connect to them. This is in stark contrast to C_1 where around half of all nodes provided only local/private IP addresses, highlighting the success of newer versions of IPFS in identifying client nodes behind NATs.

IPFS supports connections through multiple network layer protocols; Table 13 shows the prevalence of encountered protocols during our crawls. If a node was reachable through multiple, say IPv4 addresses, we only count it as one occurrence of IPv4 to not distort the count. This is aggregated over all crawls: if a peer supported, e.g., IPv6 at some time, then it will be counted as supporting IPv6 in general. The protocols "ipfs" and "p2p-circuit" are connections through IPFS' relay nodes, "dnsaddr", "dns4/6" are DNS-resolvable addresses and "onion3" signals TOR capabilities. In comparison to C_1 there are not many differences except for the decline in IPv6-enabled nodes.

Protocol	Perc. of peers	Abs. count
ip4	99.8	1.37e5
ip6	57.6	7.9e4
p2p-circuit	5.4	7e3
onion3	0.7	979
dns4	1.0	1356
p2p	< 0.1	33
dns	< 0.1	35
dnsaddr	< 0.1	27
dns6	< 0.1	9

Table 13: Network statistics: protocol capabilities among IPFS peers.

7.1.1.3 *Overlay Graph Properties*

Figures 48 and 49 depict the log-log in-degree distribution, differentiated by all and only reachable nodes, from the crawl on the 1st of February 2021, 00:00 UTC — other crawls yielded similar shapes. In contrast to the results in Nov. 2019, it can be seen that the degree distribution of all nodes (cf. Figure 48) can, by visible inspection alone, not be characterized by either a log-normal or power-law distribution. This is also underlined by the p-values for both distributions of $\ll 0.1$ — the power-law and log-normal hypotheses therefore have to be rejected [59]. Only the Poisson distribution has a p-value of > 0.1, however, as can be seen in the figures only the tail of the empirical distribution matches the Poisson distribution, effectively ruling out this hypothesis as well. The reasons for this difference may lie in the longer uptimes of nodes, as clients are now successfully excluded from the DHT, or may be due to IPFS' bucket structure, which renders sampling through crawling necessarily incomplete with regard to the number of edges. Regarding the distribution of reachable nodes in Figure 49, the analysis yields a p-value of $p = 0.26 > 0.1$ for the power-law distribution with $x_{min} = 495$. Log-normal and Poisson hypotheses have to be rejected due to their p-value being $\ll 0.1$ or simply due to a large x_{min} in the case of the Poisson distribution. Therefore, at least the tail of the degree distribution, i.e., all degrees larger than x_{min} can be plausibly characterized by a power-law.

One striking observation in comparison to C_1 is the peak in degrees of the graph of reachable nodes in the interval $[200, 230]$. Around one third of all nodes are within this degree range. This has consequences on other graph metrics but, most strikingly, on the resilience analysis as the graph of reachable nodes tolerates failures and targeted attacks surprisingly well — achieving comparable results to similar studies on Gnutella [250].

Table 14 depicts the aggregated statistics on the degree of each crawls' graph. Depicted is the mean, median, minimum and maximum degree, differentiated by in- and out-degree.

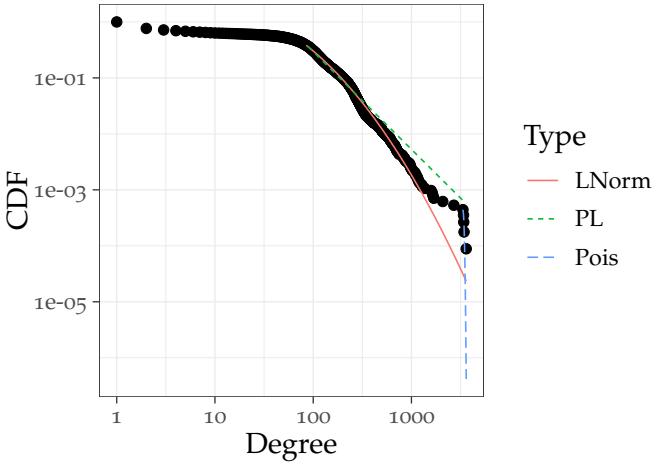


Figure 48: In-Degree distribution from the crawl on 1st of February 2021, 00:00 UTC including *all* found nodes. Other crawls yielded similar shapes.

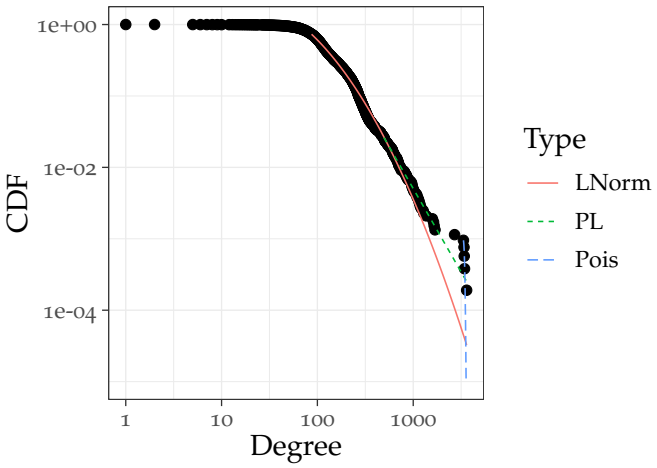


Figure 49: The same distribution as Figure 48, but only including *reachable* nodes.

The fluctuation of top degree nodes is depicted in the cumulative distribution in Figure 50. For the computation, each crawl's highest degree nodes (15 on average) were calculated and their re-occurrence counted. If a node was in the set of highest-degree nodes in one run but not in other runs, its "percentage seen" would be $\frac{1}{\#\text{ of crawls}}$. On the other extreme, if a node was within the highest degree nodes in every crawl, its percentage seen would be a 100 %. In comparison to C_1, Figure 50 shows less churn among the top degree nodes: although 60 % of top degree nodes were only present in 10 % of crawls, the remaining 40 % have been seen in the the majority of crawls. This is in line with longer observed session lengths in C_2 in general.

Similar to Section 4.7.3.2, we computed various graph metrics of the IPFS overlay graph and compared them to ER and BA random graphs

mode	min	mean	median	max
total	23.8	312.4	273.1	3748.6
in	1.1	156.2	114.1	3581.5
out	8.7	156.2	162.0	187.9

Table 14: Average of the degree statistics of all 5039 crawls.

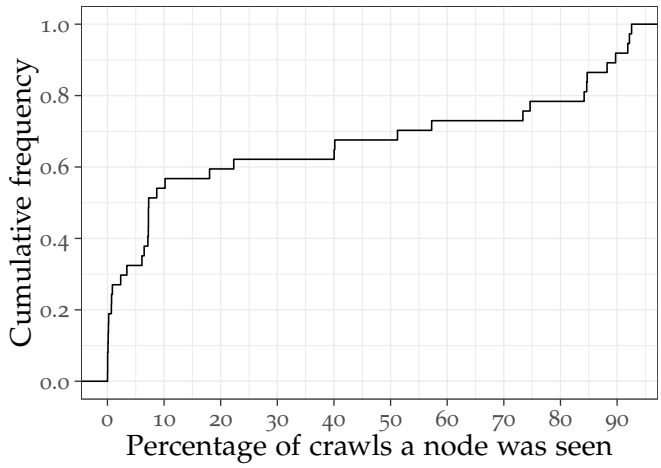


Figure 50: ECDF of how often the same nodes were within the top-degree nodes.

– the methods and parameters are identical to Table 6. As above, we used the crawl from the 1st of February 2021, 00:00 UTC, the results are gathered in Table 15. It can be seen that the IPFS graphs in C_2 have seven to eight times more edges than the equivalent graphs in C_1 despite having less nodes. This stems, among potential other reasons, from two factors: (1) significant improvements of the libp2p networking library and our crawler and (2) longer session lengths. Regarding (1), since IPFS v0.5, the Kademlia buckets are populated much more rigorously and quickly after startup. Furthermore, refreshments at least every 10 min lead to more nodes in buckets than for older IPFS clients. We also improved our crawler, especially the interplay with libp2p, leading to a higher throughput of crawl requests and more reliability. Simultaneously, session lengths have become significantly longer (clearly visible in Table 11), leading to fuller Kademlia buckets. In summary, in comparison to C_1, the crawler is able to enumerate V' evenly well but performs significantly better on E'.

It can be seen that, similar to C_1, the IPFS graphs exhibit similar average path lengths as ER random graphs, although with a higher diameter. The transitivity is, again, significantly higher than for random graphs. Assessing the small-world behavior, we obtain $S_{\text{all}} = 10 > 1$ and $S_{\text{reach}} = 2.4 > 1$, which is in the same ballpark as C_1 and still allows for the qualitative categorization as a small-world graph. Quantitatively, the metric of small-worldness S is much lower for IPFS (both

GType	Metric	All	Reach
ba	nodes	11311	5249
ba	edges	935327	801505
ba	apl	1.01±0.01	1.00
ba	trans	0.022	0.084
ba	betw	1e-04	0
ba	diameter	2.9±0.63	2.2±0.3
er	nodes	11311	5249
er	edges	939058	814201
er	apl	2.53	1.98
er	trans	0.015	0.058
er	betw	2e-04	3e-04
er	diameter	3	3
ipfs	nodes	11311	5249
ipfs	edges	939058	814201
ipfs	apl	2.58	2.24
ipfs	trans	0.152	0.161
ipfs	diameter	5	4
ipfs	betw	0.005	0.010

Table 15: Comparison of measured graphs of the IPFS overlay at 1.02.21, 00:00 UTC with Erdős-Rényi & Barabási-Albert random graphs.

sets of crawls) than for the actor collaboration network, for example [132]. In general, we can see that the IPFS graphs in C_2 are not as similar to scale-free networks as for the graphs in C_1.

Moving on to the resilience analysis (cf. Figures 51 and 52), the differences between the two sets of crawls become even more apparent. While the graph of all nodes is, not surprisingly due to the higher number of edges, more robust against failures and attacks, it still exhibits similar behavior to the equivalent graph in C_1. However, the graph of reachable nodes is remarkable in the sense that it does not partition into several connected components. Instead, despite the deletion of nodes the fraction of connected remaining nodes stays high, even when targeting the highest degree nodes. This can plausibly be explained by the high clustering coefficient and homogeneous degree distribution: neighborhoods are very well interconnected and around one third of all nodes share the same (high) degree in the interval $[200, 230]$. Hence, the difference between targeted and random removal becomes less severe. Due to the high interconnectedness of neighbors nodes tend to remain connected even when one of their neighboring nodes is removed.

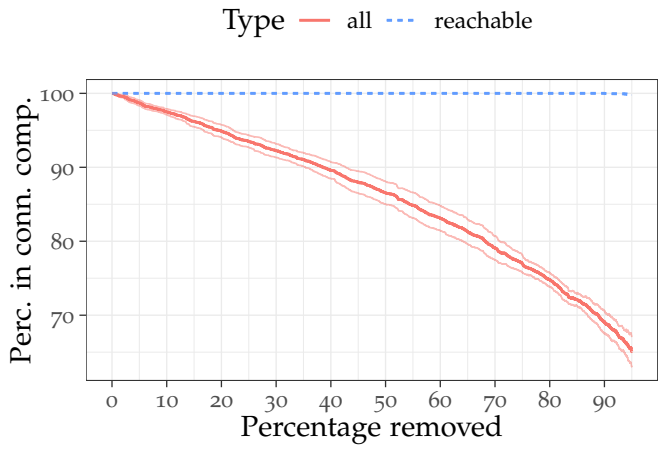


Figure 51: Resilience of the measured graph (same as above) to random removals, distinguished by all and reachable nodes.

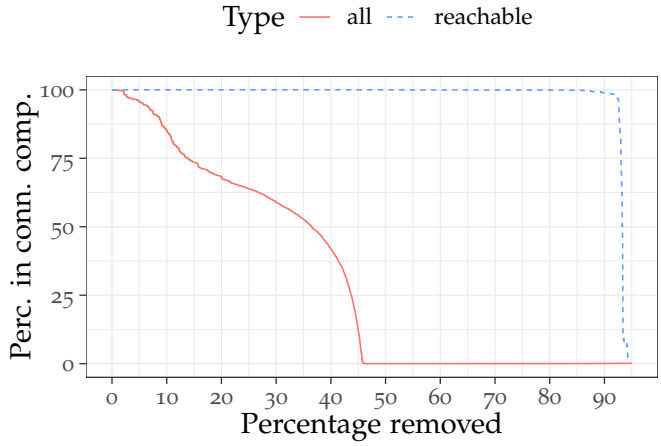


Figure 52: Resilience of the measured graph (same as above) to targeted removals, distinguished by all and reachable nodes.

7.1.2 *Gateway Probing Results*

	Results							
	2020-05-31, 10:30 AM				2020-06-08, 19:20 PM			
	de-small		us-small		de-small		us-small	
Gateway	HTTP	BitSwap	HTTP	BitSwap	HTTP	BitSwap	HTTP	BitSwap
10.via0.com					working	found	working	found
cdn.cwinfo.net								
cloudflare-ipfs.com	working	found	working	found	working	found	working	found
gateway.blocksec.com								
gateway.ipfs.io	working	found	working	found	working	found	working	found
gateway.originprotocol.com		found		found		found		found
gateway.pinata.cloud	working	found	working	found	working	found	working	found
gateway.serph.network								
gateway.temporal.cloud	working	found	working	found	working	found	working	found
hardbin.com	working	found	working	found			working	found
ipfs.2read.net	working	found	working	found	working	found	working	found
ipfs.best-practice.se	working	found	working	found	working	found	working	found
ipfs.busy.org								
ipfs.cf-ipfs.com		found		found		found		found
ipfs.cosmos-ink.net	working	found	working	found	working	found	working	found
ipfs.doolta.com								
ipfs.dweb.link	working	found	working	found	working	found	working	found
ipfs.eternum.io	working	found	working	found	working	found	working	found
ipfs.fooock.com								
ipfs.globalupload.io	working	found	working	found	working	found	working	found
ipfs.greyh.at	working	found	working	found	working	found	working	found
ipfs.infura.io	working	found	working	found	working	found	working	found
ipfs.io	working	found	working	found	working	found	working	found
ipfs.jeroendeneef.com	working	found	working	found	working	found	working	found
ipfs.lc	not listed [a]							
ipfs.leiyun.org	not listed [a]				working	found	working	[b]
ipfs.mrh.io								
ipfs.overpi.com	working	found	working	found	working	found	working	found
ipfs.privacytools.io								
ipfs.runfission.com	working	found	working	found	working	found	working	found
ipfs.sloppyta.co	working	found	working	found	working	found	working	found
ipfs.stibarc.com	working	found	working	found				
ipns.co	working	found	working	found	working	found	working	found
jorropo.ovh								
ninetailed.ninja	working	found	working	found	working	found	working	found
permaweb.io								
storjipfs-gateway.com								
trusti.id	working	found	working	found	working	found	working	found

[a] Only present in later versions of the public gateway list.
[b] Possible that the BitSwap message was missed due to backpressure in the setup.

Table 16: Results of probing public gateways. The HTTP column indicates whether we were able to request our content via HTTP using the gateway. The BitSwap column indicates whether we saw a request for our data via BitSwap.

7.2 VALUE STABILIZATION IN CRYPTOCURRENCIES

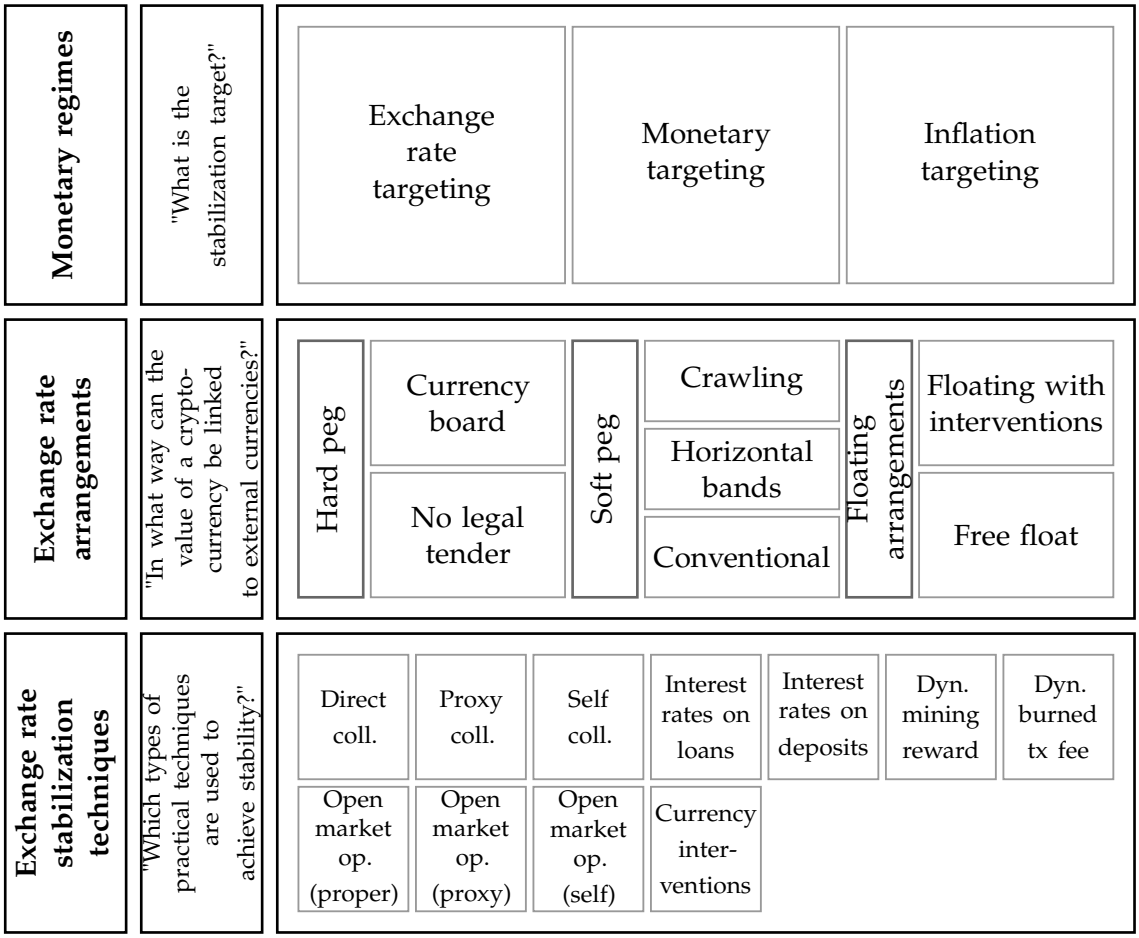
7.2.1 *Surveyed projects*

A list of surveyed stablecoin projects and their respective classification. Note that, "partially implemented" refers to the fact that the coin itself is traded while not all announced stabilization techniques are implemented yet.

Project (Stabilized Token)	Status	MR	ERA	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
Augmint (A-EUR)	not impl.	ERT	soft peg	-	-	-	-	yes	yes	-	yes	-	-	-
Aurora (Boreal)	not impl.	ERT	soft peg	-	-	-	-	yes	-	-	yes	-	-	-
Basecoin (Basis)	retracted	ERT	soft peg	-	-	-	-	-	-	-	-	yes	yes	-
BitShares (BitUSD)	impl.	ERT	soft peg	-	-	yes	-	-	-	-	-	-	-	-
Carbon (Carbon)	impl.	ERT	soft peg	-	-	-	-	-	-	-	-	yes	yes	-
Celo (Celo)	not impl.	ERT	soft peg	-	-	-	-	-	-	-	yes	yes	yes	-
Centre (USD Coin)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Digix (Digix Gold Token)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Fragments (Fragments)	not impl.	ERT	soft peg	-	-	-	-	-	-	-	yes	yes	-	-
Globcoin (GLC Token)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Karbo (Karbo)	partially impl.	MT	free float	-	-	-	-	-	yes	-	-	-	yes	-
Kowala (kUSD)	not impl.	ERT	soft peg	-	-	-	-	-	-	-	-	-	yes	yes
Maker (Dai)	impl.	ERT	soft peg	-	-	-	-	yes	-	-	-	-	-	-
Minex Coin (Minex Coin)	impl.	MT	float. w. int.	-	-	-	yes	-	yes	-	-	-	-	-
NuBits (Nubits)	impl.	ERT	soft peg	-	-	-	yes	-	yes	-	-	-	yes	-
Stableunit (Stableunit)	not impl.	ERT	soft peg	-	-	-	-	-	yes	-	yes	yes	-	-
Stably (StableUSD)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Stasis (EURS)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Stronghold (USDS)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Sythetix (SUSD)	impl.	ERT	soft peg	-	-	yes	-	-	-	-	-	-	-	-
Tether (USDT)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
Trusttoken (TrueUSD)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
USC (USC)	impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-
x8currency (X8C)	not impl.	ERT	hard peg	yes	-	-	-	-	-	-	-	-	-	-

Abbreviation	**Full text**
MR	Monetary regime
ERA	Exchange range arrangement
MT	Monetary targeting
ERT	Exchange rate targeting
float. w. int.	Floating with interventions
impl.	Implemented
T1	Collateralization (direct)
T2	Collateralization (proxy)
T3	Collateralization (self)
T4	Currency interventions
T5	Interest rates with loans
T6	Interest rates with deposits
T7	Open market operations (standard)
T8	Open market operations (proxy)
T9	Open market operations (self-tokenizing)
T10	Dynamic mining reward
T11	Dynamically burned transaction fee

7.2.2 *Taxonomy*



BIBLIOGRAPHY

PUBLICATIONS BY THE AUTHOR

[1] Leonhard Balduf, Sebastian A. Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. "Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS". In: *submitted to ICDCS* (2022).

[2] S. Beaucamp, S. Henningsen, and M. Florian. "Strafbarkeit durch Speicherung der Bitcoin-Blockchain?" In: *Multimedia und Recht* 2018.8 (Aug. 2018), pp. 493–564.

[3] Moritz Becker, Sebastian Henningsen, and Ingolf GA Pernice. "Umstrittene Expertise im Falle einer neuen Technologie: eine explorative Untersuchung der Online-Konsultation zur Blockchain-Strategie der Bundesregierung". In: *Weizenbaum Series* (2020).

[4] Martin Florian, Sebastian A. Henningsen, Sophie Beaucamp, and Björn Scheuermann. "Erasing Data from Blockchain Nodes". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P) Workshops*. IEEE, 2019, pp. 367–376. URL: https://doi.org/10.1109/EuroSPW.2019.00047.

[5] Martin Florian, Sebastian A. Henningsen, and Björn Scheuermann. "The Sum of Its Parts: Analysis of Federated Byzantine Agreement Systems". In: *CoRR* abs/2002.08101 (2020). arXiv: 2002.08101. URL: https://arxiv.org/abs/2002.08101.

[6] Sebastian A. Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. "Mapping the Interplanetary Filesystem". In: *Proceedings of the IFIP Networking Conference*. IFIP, 2020, pp. 289–297. URL: https://ieeexplore.ieee.org/document/9142766.

[7] Sebastian A. Henningsen, Sebastian Rust, Martin Florian, and Björn Scheuermann. "Crawling the IPFS Network". In: *Proceedings of IFIP Networking Conference*. IFIP, 2020, pp. 679–680. URL: https://ieeexplore.ieee.org/document/9142764.

[8] Sebastian A. Henningsen, Daniel Teunis, Martin Florian, and Björn Scheuermann. "Eclipsing Ethereum Peers with False Friends". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P) Workshops*. IEEE, 2019, pp. 300–309. URL: https://doi.org/10.1109/EuroSPW.2019.00040.

[9] Ingolf G. A. Pernice, Sebastian A. Henningsen, Roman Proskalovich, Martin Florian, Hermann Elendner, and Björn Scheuermann. "Monetary Stabilization in Cryptocurrencies - Design Approaches and Open Questions". In: *Proceedings of Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2019, pp. 47–59. URL: https://doi.org/10.1109/CVCBT.2019.00011.

OTHER PUBLICATIONS

[10] John Adler, Ryan Berryhill, Andreas G. Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. "Astraea: A Decentralized Blockchain Oracle". In: *Proceedings of the International Conference on Internet of Things (iThings) and Green Computing and Communications (GreenCom) and Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData)*. IEEE, 2018, pp. 1145–1152. URL: https://doi.org/10.1109/Cybermatics_2018.2018.00207.

[11] Bithin Alangot, Daniël Reijsbergen, Sarad Venugopalan, and Pawel Szalachowski. "Decentralized Lightweight Detection of Eclipse Attacks on Bitcoin Clients". In: *Proceedings of the Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 337–342. URL: https://doi.org/10.1109/Blockchain50366.2020.00049.

[12] Réka Albert, Hawoong Jeong, and Albert-László Barabási. "Error and attack tolerance of complex networks". In: *Nature* 406.6794 (2000), pp. 378–382.

[13] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. "IoT data privacy via blockchains and IPFS". In: *Proceedings of the Conference on the Internet of Things (IOT)*. ACM, 2017, 14:1–14:7. URL: https://doi.org/10.1145/3131542.3131563.

[14] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies". In: *ACM computing surveys (CSUR)* 36.4 (2004), pp. 335–371.

[15] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies". In: *Proceedings of Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 375–392. URL: https://doi.org/10.1109/SP.2017.29.

[16] Frederik Armknecht, Ghassan O Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. "Ripple: Overview and outlook". In: *Proceedings of the Conference on Trust and Trustworthy Computing (TRUST)*. Springer. 2015, pp. 163–180.

[17] Onur Ascigil, Sergi Reñé, Michal Król, George Pavlou, Lixia Zhang, Toru Hasegawa, Yuki Koizumi, and Kentaro Kita. "Towards Peer-to-Peer Content Retrieval Markets: Enhancing IPFS with ICN". In: *Proceedings of the Conference on Information-Centric Networking (ICN)*. ACM, 2019, pp. 78–88. URL: https://doi.org/10.1145/3357150.3357403.

[18] Elikem Attah. *Five most prolific 51% attacks in crypto: Verge, Ethereum Classic, Bitcoin Gold, Feathercoin, Vertcoin*. 2019. URL: https://cryptoslate.com/prolific-51-attacks-crypto-verge-ethereum-classic-bitcoin-gold-feathercoin-vertcoin/.

[19] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A survey of attacks on ethereum smart contracts (sok)". In: *Proceedings of the Conference on Principles of Security and Trust (POST)*. Springer, 2017, pp. 164–186. URL: https://doi.org/10.1007/978-3-662-54455-6_8.

[20] Adam Back et al. *Hashcash-a denial of service counter-measure*. 2002. URL: http://www.hashcash.org/papers/.

[21] European Central Bank. *Eligibility criteria and assessment*. 2019. URL: https://www.ecb.europa.eu/mopo/assets/html/index.en.html.

[22] European Central Bank. *Guiding principles (with examples) of Eurosystem-preferred eligible ABSs*. July 2015. URL: https://www.ecb.europa.eu/mopo/implement/omt/html/abs_guiding_principles.en.html.

[23] European Central Bank. *Purchase programme for covered bonds*. 2009. URL: https://www.ecb.europa.eu/press/pr/date/2009/html/pr090604_1.en.html.

[24] European Central Bank. *The Eurosystem's instruments*. 2019. URL: https://www.ecb.europa.eu/mopo/implement/html/index.en.html.

[25] Markets Committee of the Bank for International Settlements. *Monetary policy frameworks and central bank market operations*. Tech. rep. Bank for International Settlements, May 2009. URL: https://www.bis.org/publ/mktc04.pdf.

[26] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. "SoK: Consensus in the age of blockchains". In: *Proceedings of the Conference on Advances in Financial Technologies (AFT)*. ACM, 2019, pp. 183–198. URL: https://doi.org/10.1145/3318041.3355458.

[27] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512.

[28] Salman Baset and Henning Schulzrinne. "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol". In: *Proceedings of the Conference on Computer Communications (INFOCOM)*. IEEE, 2006. URL: https://doi.org/10.1109/INFOCOM.2006.312.

[29] Ingmar Baumgart and Sebastian Mies. "S/Kademlia: A practicable approach towards secure key-based routing". In: *Proceedings of the Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2007, pp. 1–8. URL: https://doi.org/10.1109/ICPADS.2007.4447808.

[30] Morten L. Bech and Rodney Garratt. "Central Bank Cryptocurrencies". In: *SSRN* abs/3041906 (2017). URL: https://papers.ssrn.com/abstract=3041906.

[31] Moritz Becker. "Blockchain and the Promise(s) of Decentralisation: A Sociological Investigation of the Sociotechnical Imaginaries of Blockchain". In: *Proceedings of the Conference on Critical Issues in Science, Technology and Society Studies*. STS, 2020.

[32] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, 1993, pp. 62–73. URL: https://doi.org/10.1145/168588.168596.

[33] Sami Ben Mariem, Pedro Casas, and Benoit Donnet. "Vivisecting blockchain P2P networks: Unveiling the Bitcoin IP network". In: *Proceedings of the Conference on Emerging Network Experiment and Technology (CoNEXT) Student Workshop*. ACM, 2018.

[34] Juan Benet. "IPFS - Content Addressed, Versioned, P2P File System". In: *CoRR* abs/1407.3561 (2014). arXiv: 1407.3561. URL: http://arxiv.org/abs/1407.3561.

[35] Ben S Bernanke, Thomas Laubach, Frederic S Mishkin, and Adam S Posen. *Inflation targeting*. Princeton University Press, 1999.

[36] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Keccak". In: *Proceedings of the Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2013, pp. 313–314. URL: https://doi.org/10.1007/978-3-642-38348-9_19.

[37] Mr Ashok Bhundia and Mr Mark R Stone. *A new taxonomy of monetary regimes*. Tech. rep. International Monetary Fund, 2004. URL: https://www.imf.org/en/Publications/WP/Issues/2016/12/31/A-New-Taxonomy-of-Monetary-Regimes-17690.

[38] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. "Deanonymisation of Clients in Bitcoin P2P Network". In: *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, 2014, pp. 15–29. URL: https://doi.org/10.1145/2660267.2660379.

[39] BitTorrent. *BitTorrent FileSystem (BTFS): Scalable Decentralized File Storage*. 2020. URL: https://github.com/TRON-US/go-btfs.

[40] Ole Bjerg. "Designing New Money - The Policy Trilemma of Central Bank Digital Currency". In: *SSRN* abs/2985381 (2017). URL: https://papers.ssrn.com/abstract=2985381.

[41] Charles Blake and Rodrigo Rodrigues. "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two." In: *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*. USENIX Association, 2003, pp. 1–6.

[42] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. "Complex networks: Structure and dynamics". In: *Physics reports* 424.4-5 (2006), pp. 175–308.

[43] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies". In: *Proceedings of the Symposium on Security and Privacy (SP)*. IEEE, 2015, pp. 104–121. URL: https://doi.org/10.1109/SP.2015.14.

[44] John Buford, Heather Yu, and Eng Keong Lua. *P2P networking and applications*. Morgan Kaufmann, 2009.

[45] Dirk Bullmann, Jonas Klemm, and Andrea Pinna. "In search for stability in crypto-assets: are stablecoins the solution?" In: *ECB Occasional Paper* 230 (2019).

[46] Vitalik Buterin. *vitalik.eth on Twitter*. 2018. URL: https://twitter.com/VitalikButerin/status/1051160932699770882?ref_src=twsrc%5C%5Etfw.

[47] Carey Caginalp. "A Dynamical Systems Approach to Cryptocurrency Stability". In: *CoRR* abs/1805.03143 (2018). URL: http://arxiv.org/abs/1805.03143.

[48] Carey Caginalp and Gunduz Caginalp. "Opinion: Valuation, liquidity price, and stability of cryptocurrencies". In: *Proceedings of the National Academy of Sciences* 115.6 (2018), pp. 1131–1134.

[49] Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verssimo. "Exploring the Monero Peer-to-Peer Network". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Ed. by Joseph Bonneau and Nadia Heninger. Vol. 12059. Lecture Notes in Computer Science. Springer, 2020, pp. 578–594. URL: https://doi.org/10.1007/978-3-030-51280-4_31.

[50] Miguel Castro, Peter Druschel, Ayalvadi J. Ganesh, Antony I. T. Rowstron, and Dan S. Wallach. "Secure Routing for Structured Peer-to-Peer Overlay Networks". In: *Proceedings of the Symposium on Operating System Design and Implementation (OSDI)*. USENIX Association, 2002. URL: http://www.usenix.org/events/osdi02/tech/castro.html.

[51] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of the Symposium on Operating System Design and Implementation (OSDI)*. USENIX Association, 1999, pp. 173–186. URL: https://dl.acm.org/citation.cfm?id=296824.

[52] David Chaum. "Blind Signatures for Untraceable Payments". In: *Proceedings of the Cryptology Conference (CRYPTO)*. Plenum Press, 1982, pp. 199–203. URL: https://doi.org/10.1007/978-1-4757-0602-4_18.

[53] David Chaum, Amos Fiat, and Moni Naor. "Untraceable electronic cash". In: *Proceedings of the Cryptology Conference (CRYPTO)*. Springer. 1988, pp. 319–327. URL: https://doi.org/10.1007/0-387-34799-2_25.

[54] Alice Cheng and Eric Friedman. "Sybilproof Reputation Mechanisms". In: *Proceedings of the SIGCOMM workshop on Economics of peer-to-peer systems (P2PECON)*. ACM, 2005, pp. 128–132. URL: https://doi.org/10.1145/1080192.1080202.

[55] Samuel Cheun, Isabel von Köppen-Mertes, and Benedict Weller. *The collateral frameworks of the Eurosystem, the Federal Reserve System and the Bank of England and the financial market turmoil*. Tech. rep. ECB Occasional Paper, 2009. URL: https://www.ecb.europa.eu/pub/pdf/scpops/ecbocp107.pdf?cfe69002f33a9de3dfaaad38ce139e18.

[56] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. "Evaluation of Sybil Attacks Protection Schemes in KAD". In: *Proceedings of the Conference on Scalability of Networks and Services (AIMS)*. Springer, 2009, pp. 70–82. URL: https://doi.org/10.1007/978-3-642-02627-0_6.

[57] Thibault Cholez, Isabelle Chrisment, Olivier Festor, and Guillaume Doyen. "Detection and mitigation of localized attacks in a widely deployed P2P network". In: *Peer Peer Netw. Appl.* 6.2 (2013), pp. 155–174. URL: https://doi.org/10.1007/s12083-012-0137-7.

[58] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. "Protecting Free Expression Online with Freenet". In: *IEEE Internet Comput.* 6.1 (2002), pp. 40–49. URL: https://doi.org/10.1109/4236.978368.

[59] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. "Power-Law Distributions in Empirical Data". In: *SIAM Rev.* 51.4 (2009), pp. 661–703. URL: https://doi.org/10.1137/070710111.

[60] Matt Condon. *Parity Wallet Hack 2: Electric Boogaloo*. Nov. 2017. URL: https://hackernoon.com/parity-wallet-hack-2-electric-boogaloo-e493f2365303.

[61] B. Confais, A. Lebre, and B. Parrein. "Performance Analysis of Object Store Systems in a Fog/Edge Computing Infrastructures". In: *Proceedings of the Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016, pp. 294–301. URL: https://doi.org/10.1109/CloudCom.2016.0055.

[62] Bastien Confais, Adrien Lebre, and Benot Parrein. "An Object Store Service for a Fog/Edge Computing Infrastructure Based on IPFS and a Scale-Out NAS". In: *Proceedings of the Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2017, pp. 41–50. URL: https://doi.org/10.1109/ICFEC.2017.13.

[63] Andrew Crockett et al. "Monetary policy implications of increased capital flows". In: *Changing Capital Markets: Implications for Monetary Policy*. Citeseer, 1994, pp. 331–364.

[64] Kyle Croman et al. "On Scaling Decentralized Blockchains - (A Position Paper)". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC) Workshops*. Springer, 2016, pp. 106–125. URL: https://doi.org/10.1007/978-3-662-53357-4_8.

[65] Wei Dai. *B-money*. 1998. URL: http://www.weidai.com/bmoney.

[66] Phil Daian. *Analysis of the DAO exploit*. June 2016. URL: https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/.

[67] Phil Daian, Rafael Pass, and Elaine Shi. "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Springer, 2019, pp. 23–41. URL: https://doi.org/10.1007/978-3-030-32101-7_2.

[68] George Danezis and Sarah Meiklejohn. "Centrally Banked Cryptocurrencies". In: *Proc. of Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2016.

[69] Erik Daniel, Elias Rohrer, and Florian Tschorsch. "Map-Z: Exposing the Zcash Network in Times of Transition". In: *Proceedings of the Conference on Local Computer Networks (LCN)*. IEEE, 2019, pp. 84–92. URL: https://doi.org/10.1109/LCN44214.2019.8990796.

[70] Erik Daniel and Florian Tschorsch. "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks". In: *CoRR* abs/2102.12737 (2021). arXiv: 2102.12737. URL: https://arxiv.org/abs/2102.12737.

[71] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. "Dynamo: amazon's highly available key-value store". In: *Proceedings of the Symposium on Operating Systems Principles*

(SOSP). ACM, 2007, pp. 205–220. URL: https://doi.org/10.1145/1294261.1294281.

[72] Christian Decker and Roger Wattenhofer. "Information propagation in the Bitcoin network". In: *Proceedings of the Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013, pp. 1–10. URL: https://doi.org/10.1109/P2P.2013.6688704.

[73] Sergi Delgado-Segura, Surya Bakshi, Cristina Perez-Sola, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. "TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Springer, 2019, pp. 550–566. URL: https://doi.org/10.1007/978-3-030-32101-7_32.

[74] Aragon Developers. *Aragon DAO Whitepaper*. 2020. URL: https://github.com/aragon/whitepaper/tree/1b8d87ab7e41a368d74a39320ec63a26535ecb03.

[75] Bitcoin Developers. *Bitcoind*. https://github.com/bitcoin/bitcoin. 2020. URL: https://github.com/bitcoin/bitcoin/blob/d8ca51db5ddfb51c225c0c99f8aa1f5d68f0ad83/src/addrman.h%5C#L97.

[76] Bitcoin Developers. *Bitcoind*. https://github.com/bitcoin/bitcoin. 2020. URL: https://github.com/bitcoin/bitcoin/blob/551dc7f664666cdc8cb6e6cab5522d70980778e8/src/netaddress.cpp#L528.

[77] Bitcoin Developers. *Bitcoind*. https://github.com/bitcoin/bitcoin. 2020. URL: https://github.com/bitcoin/bitcoin/blob/9fb95ae8e3e4f10888a98fc99d704d97e2eb371f/src/net.cpp#L850.

[78] Bitcoin Developers. *PR #7840: Several performance and privacy improvements to inv/mempool handling*. 2016. URL: https://github.com/bitcoin/bitcoin/pull/7840/commits.

[79] Bitcoin Cash Developers. *Bitcoin Cash*. 2017. URL: https://www.bitcoincash.org/.

[80] Bitcoin Gold Developers. *Bitcoin Gold*. 2017. URL: https://bitcoingold.org/.

[81] Blockchain.com Developers. *Orphaned Blocks Statistic*. 2020. URL: https://www.blockchain.com/en/charts/n-orphaned-blocks.

[82] Microsoft Developers. *Microsoft ION: DID Method implementation using the Sidetree protocol on top of Bitcoin*. 2020. URL: https://github.com/decentralized-identity/ion.

[83] Pinata Developers. *Pinata — Add files to IPFS effortlessly*. 2021. URL: https://pinata.cloud/.

[84] Lars Dittmar and Aaron Praktiknjo. "Could Bitcoin emissions push global warming above 2°C?" In: *Nature Climate Change* 9.9 (2019), pp. 656–657.

[85] Trinh Viet Doan, Tat Dat Pham, Markus Oberprieler, and Vaibhav Bajpai. "Measuring Decentralized Video Streaming: A Case Study of DTube". In: *Proceedings of the IFIP Networking Conference*. IFIP, 2020, pp. 118–126. URL: https://ieeexplore.ieee.org/document/9142739.

[86] John R Douceur. "The sybil attack". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2002, pp. 251–260. URL: https://doi.org/10.1007/3-540-45748-8_24.

[87] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. "Inside dropbox: understanding personal cloud storage services". In: *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2012, pp. 481–494. URL: https://doi.org/10.1145/2398776.2398827.

[88] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[89] Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. "Practical attacks against the I2P network". In: *Proceedings of the Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Springer, 2013, pp. 432–451. URL: https://doi.org/10.1007/978-3-642-41284-4_22.

[90] Barry Eichengreen. "The EMS crisis in retrospect". In: *SSRN* abs/264347 (2000). URL: https://papers.ssrn.com/abstract_id=253143.

[91] Barry Eichengreen and Ricardo Hausmann. "Exchange rates and financial fragility". In: *Proceedings of the Symposium on Economic Policy*. 1999, pp. 329–368.

[92] Barry Eichengreen, Andrew K Rose, and Charles Wyplosz. *Speculative attacks on pegged exchange rates: an empirical exploration with special reference to the European Monetary System*. 1994. URL: https://www.nber.org/papers/w4898.

[93] Paul Erds and Alfréd Rényi. "On the evolution of random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[94] European Union. *Directive 2000/31/EC, 'Directive on electronic commerce'*. June 2000. URL: https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX%3A32000L0031.

[95] Blockchains Expert. *Decentralized Applications Stack*. 2018. URL: https://hackernoon.com/decentralized-applications-stack-3ac9ab8c4ad7.

[96] Vitalik Buterin Fabian Vogelsteller. *Ethereum Improvement Proposal #20*. Nov. 2015. URL: https://eips.ethereum.org/EIPS/eip-20.

[97] Romano Fantacci, Leonardo Maccari, Matteo Rosi, Luigi Chisci, Luca Maria Aiello, and Marco Milanesio. "Avoiding Eclipse Attacks on Kad/Kademlia: An Identity Based Approach". In: *Proceedings of Conference on Communications (ICC)*. IEEE, 2009, pp. 1–5. URL: https://doi.org/10.1109/ICC.2009.5198772.

[98] Giulia C. Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. "Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees". In: *Proc. ACM Meas. Anal. Comput. Syst.* 2.2 (2018), 29:1–29:35. URL: https://doi.org/10.1145/3224424.

[99] Giulia C. Fanti and Pramod Viswanath. "Anonymity Properties of the Bitcoin P2P Network". In: *CoRR* abs/1703.08761 (2017). arXiv: 1703.08761. URL: http://arxiv.org/abs/1703.08761.

[100] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering - Design Principles and Practical Applications*. Wiley, 2010. URL: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470474246.html.

[101] Marco Ferrante and Monica Saltalamacchia. "The coupon collector's problem". In: *Materials matemàtics* (2014), pp. 1–35. URL: http://mat.uab.cat/matmat_antiga/PDFv2014/v2014n02.pdf.

[102] Stanley Fischer. "Exchange rate regimes: is the bipolar view correct?" In: *Journal of economic perspectives* 15.2 (2001), pp. 3–24.

[103] Irving Fisher. "The Equation of Exchange 1896-1910". In: *The American Economic Review* 1.2 (1911), pp. 296–305.

[104] Ethereum Foundation. *Ewasm Design Overview and Specification*. 2019. URL: https://github.com/ewasm/design.

[105] Ethereum Foundation. *Go Ethereum – Official Go implementation of the Ethereum protocol*. 2020. URL: https://github.com/ethereum/go-ethereum.

[106] Ethereum Foundation. *Swarm – Storage and Communication for a Sovereign Digital Society*. 2020. URL: https://ethersphere.github.io/swarm-home/.

[107] Jeffrey A Frankel. "No single currency regime is right for all countries or at all times". In: *Essays in International Finance* 215 (1998).

[108] Christine Fricker, Philippe Robert, and James Roberts. "A versatile and accurate approximation for LRU cache performance". In: *Proceedings of the International Teletraffic Congress (ITC)*. IEEE, 2012, pp. 1–8. URL: http://ieeexplore.ieee.org/document/6331818/.

[109] Milton Friedman. "Quantity theory of money". In: *The new Palgrave dictionary of economics* (2017), pp. 1–31.

[110] International Monetary Fund. *Exchange arrangements and exchange restrictions: annual report*. Tech. rep. International Monetary Fund, 2016.

[111] Yue Gao, Jinqiao Shi, Xuebin Wang, Qingfeng Tan, Can Zhao, and Zelin Yin. "Topology Measurement and Analysis on Ethereum P2P Network". In: *Proceedings of the Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7. URL: https://doi.org/10.1109/ISCC47284.2019.8969695.

[112] P. Gai, A. Kiayias, and A. Russell. "Stake-Bleeding Attacks on Proof-of-Stake Blockchains". In: *Proceedings of the Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 85–92. URL: https://doi.org/10.1109/CVCBT.2018.00015.

[113] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. "Decentralization in Bitcoin and Ethereum Networks". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Springer, 2018, pp. 439–457. URL: https://doi.org/10.1007/978-3-662-58387-6_24.

[114] James E Gentle. *Computational statistics*. Springer, 2009.

[115] Daniel Germanus, Stefanie Roos, Thorsten Strufe, and Neeraj Suri. "Mitigating Eclipse attacks in Peer-To-Peer networks". In: *Proceedings of the Conference on Communications and Network Security (CNS)*. IEEE, 2014, pp. 400–408. URL: https://doi.org/10.1109/CNS.2014.6997509.

[116] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. "On the Security and Performance of Proof of Work Blockchains". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 2016, pp. 3–16. URL: https://doi.org/10.1145/2976749.2978341.

[117] Arthur Gervais and Karl Wüst. *Ethereum Eclipse Attacks*. ETH Zurich, 2016. URL: http://hdl.handle.net/20.500.11850/121310 (visited on 12/11/2018).

[118] Edgar N Gilbert. "Random graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.

[119] Colin S Gillespie. "Fitting heavy tailed distributions: the poweRlaw package". In: *CoRR* abs/1407.3492 (2014). arXiv: 1407.3492. URL: https://arxiv.org/abs/1407.3492.

[120] Florian Glaser, Kai Zimmermann, Martin Haferkorn, Moritz Weber, and Michael Siering. "Bitcoin – asset or currency? revealing users' hidden intentions". In: *SSRN* abs/2425247 (2014). URL: https://papers.ssrn.com/abstract_id=2425247.

[121] Morris Goldstein. *Managed floating plus. policy analyses in international economics*. Peterson Institute for International Economics, 2002.

[122] Board of Governors of the Federal Reserve System. *Policy Tools*. Feb. 2017. URL: https://www.federalreserve.gov/monetarypolicy/policytools.htm.

[123] John M. Griffin and Amin Shams. "Is Bitcoin Really Un-Tethered?" In: *SSRN* abs/3195066 (2018). URL: https://papers.ssrn.com/abstract=3195066.

[124] Matthias Grundmann, Till Neudecker, and Hannes Hartenstein. "Exploiting Transaction Accumulation and Double Spends for Topology Inference in Bitcoin". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC) Workshops*. Springer, 2018, pp. 113–126. URL: https://doi.org/10.1007/978-3-662-58820-8_9.

[125] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. "SoK: Layer-Two Blockchain Protocols". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 201–226. URL: https://doi.org/10.1007/978-3-030-51280-4_12.

[126] Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. "The Decentralized Financial Crisis". In: *Proceedings of the Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2020, pp. 1–15. URL: https://doi.org/10.1109/CVCBT50464.2020.00005.

[127] Saikat Guha, Neil Daswani, and Ravi Jain. "An Experimental Study of the Skype Peer-to-Peer VoIP System". In: *Proceedings of the Workshop on Peer-To-Peer Systems (IPTPS)*. Springer, 2006. URL: http://www.iptps.org/papers-2006/Guha-skype06.pdf.

[128] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. "A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems". In: *Comput. Commun. Rev.* 32.1 (2002), p. 82. URL: https://doi.org/10.1145/510726.510756.

[129] Stuart Haber and W Scott Stornetta. "How to time-stamp a digital document". In: *J. Cryptol.* 3.2 (1991), pp. 99–111. URL: https://doi.org/10.1007/BF00196791.

[130] Adam Harmes. "The trouble with hedge funds". In: *Review of Policy Research* 19.1 (2002), pp. 156–176.

[131] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. "Eclipse Attacks on Bitcoin's Peer-to-Peer Network". In: *Proceedings of the USENIX Security Symposium*. USENIX Association, 2015, pp. 129–144. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman.

[132] Mark D Humphries and Kevin Gurney. "Network small-worldness: a quantitative method for determining canonical network equivalence". In: *PloS one* 3.4 (2008).

[133] *Icon "Alice" & "Carol" made by xnimrodx*. URL: www.flaticon.com.

[134] *Icon "Bob" made by Payungkead*. URL: www.flaticon.com.

[135] Deloitte Insights. *Deloitte's 2019 Global Blockchain Survey – Blockchain gets down to business*. 2019. URL: https://www2.deloitte.com/content/dam/Deloitte/se/Documents/risk/DI_2019-global-blockchain-survey.pdf.

[136] Hatem Ismail, Daniel Germanus, and Neeraj Suri. "Detecting and Mitigating P2P Eclipse Attacks". In: *Proceedings of the Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015, pp. 224–231. URL: https://doi.org/10.1109/ICPADS.2015.36.

[137] Mitsuru Iwamura, Yukinobu Kitamura, Tsutomu Matsumoto, and Kenji Saito. "Can We Stabilize the Price of a Cryptocurrency?: Understanding the Design of Bitcoin and Its Potential to Compete with Central Bank Money". In: *SSRN* abs/2519367 (2014). URL: https://papers.ssrn.com/abstract=2519367.

[138] Marc Jansen, Farouk Hdhili, Ramy Gouiaa, and Ziyaad Qasem. "Do Smart Contract Languages Need to Be Turing Complete?" In: *Proceedings of the Congress on Blockchain and Applications (BLOCKCHAIN)*. Springer, 2019, pp. 19–26. URL: https://doi.org/10.1007/978-3-030-23813-1_3.

[139] Tim Jordan. *Cyberpower: The culture and politics of cyberspace and the Internet*. Psychology Press, 1999.

[140] Wall Street Journal. *A Flood of Questionable Cryptocurrency Offerings*. 2018. URL: https://www.wsj.com/graphics/whitepapers/.

[141] K. Junemann, P. Andelfinger, and H. Hartenstein. "Towards a Basic DHT Service: Analyzing Network Characteristics of a Widely Deployed DHT". In: *Proceedings of Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2011, pp. 1–7. URL: https://doi.org/10.1109/ICCCN.2011.6005906.

[142] Henrik Karlstrøm. "Do libertarians dream of electric coins? The material embeddedness of Bitcoin". In: *Distinktion: Scandinavian Journal of Social Theory* 15.1 (2014), pp. 23–36.

[143] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. "Measuring Ethereum Network Peers". In: *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2018, pp. 91–104. URL: https://dl.acm.org/citation.cfm?id=3278542.

[144] Seoyoung Kim, Atulya Sarin, and Daljeet Virdi. "Crypto-Assets Unencrypted". In: *SSRN* abs/3117859 (2018). URL: https://papers.ssrn.com/abstract_id=3117859.

[145] Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun-You Liu, and Andreea Minca. "Stablecoins 2.0: Economic Foundations and Risk-based Models". In: *Proceedings of the Conferece on Advances in Financial Technologies (AFT)*. ACM, 2020, pp. 59–79. URL: https://doi.org/10.1145/3419614.3423261.

[146] Ariah Klages-Mundt and Andreea Minca. "(In) Stability for the Blockchain: Deleveraging Spirals and Stablecoin Attacks". In: *CoRR* abs/1906.02152 (2019). arXiv: 1906.02152. URL: https://arxiv.org/abs/1906.02152.

[147] Michael Kohnen, Mike Leske, and Erwin P. Rathgeb. "Conducting and Optimizing Eclipse Attacks in the Kad Peer-to-Peer Network". In: *Proceedings of the IFIP Networking Conference*. Springer, 2009, pp. 104–116. URL: https://doi.org/10.1007/978-3-642-01399-7_9.

[148] JP Koning. "Fedcoin: a central bank-issued cryptocurrency". In: *R3 Report* 15 (2016).

[149] Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Kenneth P. Birman, and Alan J. Demers. "Active and passive techniques for group size estimation in large-scale and dynamic distributed systems". In: *J. Syst. Softw.* 80.10 (2007), pp. 1639–1658. URL: https://doi.org/10.1016/j.jss.2007.01.014.

[150] Max J Krause and Thabet Tolaymat. "Quantification of energy and carbon costs for mining cryptocurrencies". In: *Nature Sustainability* 1.11 (2018), pp. 711–718.

[151] Paul Krugman. "A model of balance-of-payments crises". In: *Journal of money, credit and banking* 11.3 (1979), pp. 311–325.

[152] Rakesh Kumar and Keith W Ross. "Peer-assisted file distribution: The minimum distribution time". In: *Proceedings of the Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2006, pp. 1–11. URL: https://doi.org/10.1109/HOTWEB.2006.355259.

[153] Parity Labs. *Parity/OpenEthereum Client*. 2020. URL: https://github.com/openethereum/openethereum.

[154] Protocol Labs. *Filecoin: A decentralized network storage for humanity's most important information*. 2020. URL: https://filecoin.io/.

[155] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine Generals Problem". In: vol. 4. 3. ACM, 1982, pp. 382–401. URL: http://doi.acm.org/10.1145/357172.357176.

[156] Felipe Larraín and Andrés Velasco. "How should emerging economies float their currencies?" In: *Economics of Transition* 10.2 (2002), pp. 365–392.

[157] Lawrence Lessig. *Code: And other laws of cyberspace*. ReadHowYouWant. com, 2009.

[158] Andrew T Levin, Volker Wieland, and John Williams. "Robustness of simple monetary policy rules under model uncertainty". In: *Monetary policy rules*. University of Chicago Press, 1999, pp. 263–318.

[159] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and Robert H Deng. "CrowdBC: A blockchain-based decentralized framework for crowdsourcing". In: *Trans. Parallel Distrib. Syst.* 30.6 (2018), pp. 1251–1266. URL: https://doi.org/10.1109/TPDS.2018.2881735.

[160] Bin Liu, Xiao Liang Yu, Shiping Chen, Xiwei Xu, and Liming Zhu. "Blockchain based data integrity service framework for IoT data". In: *Proceedings of the Conference on Web Services (ICWS)*. IEEE, 2017, pp. 468–475. URL: https://doi.org/10.1109/ICWS.2017.54.

[161] Thomas Locher, David Mysicka, Stefan Schmid, and Roger Wattenhofer. "Poisoning the Kad Network". In: *Proceedings of the Distributed Computing and Networking Conference (ICDCN)*. ACM, 2010, pp. 195–206. URL: https://doi.org/10.1007/978-3-642-11322-2_22.

[162] Marta Lokhava, Giuliano Losa, David Mazières, Graydon Hoare, Nicolas Barry, Eli Gafni, Jonathan Jove, Rafa Malinowsky, and Jed McCaleb. "Fast and secure global payments with Stellar". In: *Proceedings of the Symposium on Operating Systems Principles (SOSP)*. ACM, 2019, pp. 80–96. URL: https://doi.org/10.1145/3341301.3359636.

[163] Eric Lombrozo, Johnson Lau, and Pieter Wuille. "Segregated witness (consensus layer)". In: *Bitcoin Core Develop. Team, Tech. Rep. BIP* 141 (2015).

[164] Thomas Mager, Ernst Biersack, and Pietro Michiardi. "A measurement study of the Wuala on-line storage service". In: *Proceedings of the Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2012, pp. 237–248. URL: https://doi.org/10.1109/P2P.2012.6335804.

[165] Peter Mahlmann and Christian Schindelhauer. *Peer-to-peer-netzwerke: Algorithmen und Methoden*. Springer-Verlag, 2007.

[166] Dahlia Malkhi and Michael K. Reiter. "Byzantine Quorum Systems". In: *Distributed Comput.* 11.4 (1998), pp. 203–213. URL: https://doi.org/10.1007/s004460050050.

[167] Nathan Mantel and Bernard S Pasternack. "A class of occupancy problems". In: *The American Statistician* 22.2 (1968), pp. 23–24.

[168] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. "Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network". In: *IACR* 236 (2018). URL: http://eprint.iacr.org/2018/236.

[169] Henri Massias, X Serret Avila, and J-J Quisquater. "Design of a secure timestamping service with minimal trust requirement". In: *Proceedings of the Symposium on Information Theory in the Benelux*. IEEE, 1999.

[170] Petar Maymounkov and David Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2002, pp. 53–65. URL: https://doi.org/10.1007/3-540-45748-8_5.

[171] David Mazières. "Self-certifying file system". PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 2000. URL: http://hdl.handle.net/1721.1/86610.

[172] Bennet T McCallum. "Robustness properties of a rule for monetary policy". In: *Carnegie-Rochester conference series on public policy*. Vol. 29. Elsevier. 1988, pp. 173–203.

[173] Bennett T McCallum. *Credibility and monetary policy*. 1984.

[174] Bennett T McCallum. "Issues in the design of monetary policy rules". In: *Handbook of macroeconomics* 1 (1999), pp. 1483–1530.

[175] Michael McLeay, Amar Radia, and Ryland Thomas. "Money creation in the modern economy". In: *Bank of England Quarterly Bulletin Q1* (Mar. 2014). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2416234.

[176] Ghulam Memon, Reza Rejaie, Yang Guo, and Daniel Stutzbach. "Large-scale monitoring of DHT traffic". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2009, p. 11. URL: http://www.iptps.org/papers-2009/memon.pdf.

[177] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function". In: *Proceedings of the Cryptology Conference (CRYPTO)*. 1987, pp. 369–378. URL: https://doi.org/10.1007/3-540-48184-2_32.

[178] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. "Discovering bitcoin's public topology and influential nodes". In: (2015).

[179] Misc. *Augmints Whitepaper*. https://docs.google.com/document/d/1IQwGEsImpAv2Nlz5IgU_iCJkEqlM2VUHf5SFkcvb80A/edit. visited on 2018-07-16. 2018.

[180] Misc. *Basis Coin Whitepaper*. https://www.basis.io/basis_whitepaper_en.pdf. visited on 2018-07-13. 2017.

[181] Misc. *Bitcoin congestion over time*. 2020. URL: https://charts.woobull.com/bitcoin-congestion/.

[182] Misc. *Bitshares Whitepaper*. visited on 2018-07-16. 2015. URL: https://web.archive.org/web/20170822062343/http://docs.bitshares.eu:80/_downloads/bitshares-financial-platform.pdf.

[183] Misc. *Carbon Whitepaper*. https://www.carbon.money/whitepaper.pdf. visited on 2018-07-16. 2018.

[184] Misc. *Celo Whitepaper*. Whitepaper classified Please request access via community@celo.org. visited on 2018-11-30.

[185] Misc. *Dai Whitepaper*. https://makerdao.com/whitepaper. visited on 2018-11-30. 2017.

[186] Misc. *Fragments Whitepaper*. https://fragments.network/fragments-platform-whitepaper.pdf. visited on 2018-09-03.

[187] Misc. *MinexCoin Whitepaper*. https://minexcoin.com/whitepaper. visited on 2018-09-07.

[188] Misc. *NuBits Whitepaper*. https://www.nubits.com/about/white-paper. visited on 2018-07-13. 2014.

[189] Misc. *StableUnit Whitepaper*. https://stableunit.org/documents/StableUnit-whitepaper.pdf. visited on 2018-09-03.

[190] Misc. *Stably Whitepaper*. https://s3.ca-central-1.amazonaws.com/stably-public-documents/whitepapers/Stably+Whitepaper+v6.pdf. visited on 2018-07-16. 2018.

[191] Misc. *Stasis Whitepaper*. https://www.docdroid.net/QdCqGO9/stasis-white-paper-2.pdf. visited on 2018-07-16.

[192] Misc. *Synthetix Whitepaper*. https://havven.io/uploads/havven_whitepaper.pdf. visited on 2018-07-16. 2018.

[193] Misc. *Tether Whitepaper*. https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf. visited on 2018-07-16.

[194] Misc. *Trusttoken Whitepaper*. https://thetoken.io/TKN-White Paper-en_US.pdf.

[195] Frederic S Mishkin. "International experiences with different monetary policy regimes". In: *Journal of monetary economics* 43.3 (1999), pp. 579–605.

[196] Frederic S Miskin. *Monetary Policy Strategy*. MIT Press, 2006.

[197] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. "AP3: cooperative, decentralized anonymous communication". In: *Proceedings of the ACM SIGOPS European Workshop*. ACM, 2004, p. 30. URL: https://doi.org/10.1145/1133572.1133578.

[198] Makiko Mita, Kensuke Ito, Shohei Ohsawa, and Hideyuki Tanaka. "What is Stablecoin?: A Survey on Price Stabilization Mechanisms for Decentralized Payment Systems". In: *CoRR* abs/1906.06037 (2019). arXiv: 1906.06037. URL: http://arxiv.org/abs/1906.06037.

[199] Amani Moin, Kevin Sekniqi, and Emin Gün Sirer. "SoK: A Classification Framework for Stablecoin Designs". In: *Proceedings of the Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 174–197. URL: https://doi.org/10.1007/978-3-030-51280-4_11.

[200] Janet Morahan-Martin and Phyllis Schumacher. "Incidence and correlates of pathological Internet use among college students". In: *Computers in human behavior* 16.1 (2000), pp. 13–29.

[201] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. "A survey on essential components of a self-sovereign identity". In: *Comput. Sci. Rev.* 30 (2018), pp. 80–86. URL: https://doi.org/10.1016/j.cosrev.2018.10.002.

[202] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[203] Satoshi Nakamoto. *Mailinglist entry: Bitcoin P2P e-cash paper*. Nov. 2009. URL: https://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html.

[204] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. "Erlay: Efficient Transaction Relay for Bitcoin". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 817–831. URL: https://doi.org/10.1145/3319535.3354237.

[205] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. "Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 305–320. URL: https://doi.org/10.1109/EuroSP.2016.32.

[206] Raiden Network. *What is the raiden network*. 2018. URL: https://raiden.network/.

[207] T. Neudecker, P. Andelfinger, and H. Hartenstein. "Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network". In: *Proceedings of the Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 2016, pp. 358–367. URL: https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0070.

[208] Till Neudecker and Hannes Hartenstein. "Network Layer Aspects of Permissionless Blockchains". In: *IEEE Commun. Surv. Tutorials* 21.1 (2019), pp. 838–857. URL: https://doi.org/10.1109/COMST.2018.2852480.

[209] N Nizamuddin, K Salah, M Ajmal Azad, Junaid Arshad, and MH Rehman. "Decentralized document version control using ethereum blockchain and IPFS". In: *Computers & Electrical Engineering* 76 (2019), pp. 183–197.

[210] Russell O'Connor. "Simplicity: A New Language for Blockchains". In: *Proceedings of the Workshop on Programming Languages and Analysis for Security (PLAS@CCS)*. ACM, 2017, pp. 107–120. URL: https://doi.org/10.1145/3139337.3139340.

[211] Maurice Obstfeld. "Models of currency crises with self-fulfilling features". In: *European economic review* 40.3-5 (1996), pp. 1037–1047.

[212] Maurice Obstfeld and Kenneth Rogoff. "The mirage of fixed exchange rates". In: *Journal of Economic perspectives* 9.4 (1995), pp. 73–96.

[213] Eunseuk Oh and Jianer Chen. "Parallel Routing in Hypercube Networks with Faulty Nodes". In: *Proceedings of the Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2001, pp. 338–345. URL: https://doi.org/10.1109/ICPADS.2001.934838.

[214] Daniel Pérez and Benjamin Livshits. "Smart Contract Vulnerabilities: Does Anyone Care?" In: *CoRR* abs/1902.06710 (2019). arXiv: 1902.06710. URL: http://arxiv.org/abs/1902.06710.

[215] Jack Peterson and Joseph Krug. "Augur: a decentralized, open-source platform for prediction markets". In: *CoRR* abs/1501.01042 (2015). URL: http://arxiv.org/abs/1501.01042.

[216] Joseph Poon and Thaddeus Dryja. "The bitcoin lightning network: Scalable off-chain instant payments". In: (2016). URL: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf.

[217] Johan A. Pouwelse, Pawel Garbacki, Dick H. J. Epema, and Henk J. Sips. "The Bittorrent P2P File-Sharing System: Measurements and Analysis". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2005, pp. 205–216. URL: https://doi.org/10.1007/11558989_19.

[218] Diem Project. 2020. URL: https://wp.diem.com/en-US/wp-content/uploads/sites/23/2020/04/Libra_WhitePaperV2_April2020.pdf.

[219] ProtocolLabs. *IPFS Public Gateway Checker & Curated Gateway List*. 2020. URL: https://ipfs.github.io/public-gateway-checker/.

[220] Bernd Prünster, Alexander Marsalek, and Thomas Zefferer. "Total Eclipse of the Heart - Disrupting the InterPlanetary File System". In: *CoRR* abs/2011.00874 (2020). arXiv: 2011.00874. URL: https://arxiv.org/abs/2011.00874.

[221] Bernd Prünster, Dominik Ziegler, Christian Kollmann, and Bojan Suzic. "A Holistic Approach Towards Peer-to-Peer Security and Why Proof of Work Won't Do". In: *Proceedings of the Conference on Security and Privacy in Communication Networks (SecureComm)*. Springer, 2018, pp. 122–138. URL: https://doi.org/10.1007/978-3-030-01704-0_7.

[222] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. "Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit". In: *CoRR* abs/2003.03810 (2020). arXiv: 2003.03810. URL: https://arxiv.org/abs/2003.03810.

[223] Reza Rahimian, Shayan Eskandari, and Jeremy Clark. "Resolving the Multiple Withdrawal Attack on ERC20 Tokens". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P) Workshops*. IEEE, 2019, pp. 320–329.

[224] Anomali Threat Research. *The InterPlanetary Storm: New Malware in Wild Using InterPlanetary File Systems (IPFS) p2p network*. 2020. URL: https://www.anomali.com/blog/the-interplanetary-storm-new-malware-in-wild-using-interplanetary-file-systems-ipfs-p2p-network.

[225] Antoine Riard and Gleb Naumenko. "Time-Dilation Attacks on the Lightning Network". In: *CoRR* abs/2006.01418 (2020). eprint: 2006.01418. URL: https://arxiv.org/abs/2006.01418.

[226] Michael Rodler, Wenting Li, Ghassan O. Karame, and Lucas Davi. "Sereum: Protecting Existing Smart Contracts Against Re-Entrancy Attacks". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2019. URL: https://www.ndss-symposium.org/ndss-paper/ser

eum-protecting-existing-smart-contracts-against-re-entrancy-attacks/.

[227] Rodrigo Rodrigues and Barbara Liskov. "High Availability in DHTs: Erasure Coding vs. Replication". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2005, pp. 226–239. URL: https://doi.org/10.1007/11558989_21.

[228] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. "Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P) Workshops*. IEEE, 2019, pp. 347–356. URL: https://doi.org/10.1109/EuroSPW.2019.00045.

[229] Stefanie Roos, Benjamin Schiller, Stefan Hacker, and Thorsten Strufe. "Measuring Freenet in the Wild: Censorship-Resilience under Observation". In: *Proceedings of the Symposium on Privacy Enhancing Technologies (PETS)*. 2014, pp. 263–282. URL: https://doi.org/10.1007/978-3-319-08506-7_14.

[230] Meni Rosenfeld. "Analysis of Hashrate-Based Double Spending". In: *CoRR* abs/1402.2009 (2014). URL: http://arxiv.org/abs/1402.2009.

[231] Kenji Saito and Mitsuru Iwamura. "How to Make a Digital Currency on a Blockchain Stable". In: *CoRR* abs/1801.06771 (2018). URL: http://arxiv.org/abs/1801.06771.

[232] Hani Salah, Stefanie Roos, and Thorsten Strufe. "Characterizing graph-theoretic properties of a large-scale DHT: Measurements vs. simulations". In: *Proceedings of the Symposium on Computers and Communications (ISCC)*. IEEE, 2014, pp. 1–7. URL: https://doi.org/10.1109/ISCC.2014.6912540.

[233] Hani Salah and Thorsten Strufe. "Capturing Connectivity Graphs of a Large-Scale P2P Overlay Network". In: *Proceedings of the Conference on Distributed Computing Systems (ICDCS) Workshops*. IEEE, 2013, pp. 172–177. URL: https://doi.org/10.1109/ICDCSW.2013.35.

[234] João Santos, Nuno Santos, and David Dias. "DClaims: A Censorship Resistant Web Annotations System using IPFS and Ethereum". In: *CoRR* abs/1912.03388 (2019). arXiv: 1912.03388. URL: http://arxiv.org/abs/1912.03388.

[235] Flora Rheta Schreiber. *Sybil*. Warner Books, Inc., 1975.

[236] David Schwartz, Noah Youngs, Arthur Britto, et al. "The ripple protocol consensus algorithm". In: *Ripple Labs Inc White Paper* 5.8 (2014).

[237] Jiajie Shen, Yi Li, Yangfan Zhou, and Xin Wang. "Understanding I/O performance of IPFS storage: a client's perspective". In: *Proceedings of the Symposium on Quality of Service (IWQoS)*. ACM, 2019, 17:1–17:10. URL: https://doi.org/10.1145/3326285.3329052.

[238] Atul Singh, Miguel Castro, Peter Druschel, and Antony I. T. Rowstron. "Defending against eclipse attacks on overlay networks". In: *Proceedings of the ACM SIGOPS European Workshop*. ACM, 2004, p. 21. URL: https://doi.org/10.1145/1133572.1133613.

[239] Atul Singh, Tsuen-Wan Ngan, Peter Druschel, and Dan S. Wallach. "Eclipse Attacks on Overlay Networks: Threats and Defenses". In: *Proceedings of the Conference on Computer Communications (INFOCOM)*. IEEE, 2006. URL: https://doi.org/10.1109/INFOCOM.2006.231.

[240] Emil Sit and Robert Tappan Morris. "Security Considerations for Peer-to-Peer Distributed Hash Tables". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2002, pp. 261–269. URL: https://doi.org/10.1007/3-540-45748-8_25.

[241] Jakub Sliwinski and Roger Wattenhofer. "ABC: Asynchronous Blockchain without Consensus". In: *CoRR* abs/1909.10926 (2019). arXiv: 1909.10926. URL: http://arxiv.org/abs/1909.10926.

[242] Wolfgang Stadje. "The collector's problem with group drawings". In: *Advances in Applied Probability* (1990), pp. 866–882.

[243] Moritz Steiner, Ernst W. Biersack, and Taoufik En-Najjary. "Actively Monitoring Peers in KAD". In: *Proceedings of the Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2007. URL: http://www.iptps.org/papers-2007/SteinerBiersackEnnajjary.pdf.

[244] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. "A global view of kad". In: *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2007, pp. 117–122. URL: https://doi.org/10.1145/1298306.1298323.

[245] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. "Exploiting KAD: possible uses and misuses". In: *Comput. Commun. Rev.* 37.5 (2007), pp. 65–70. URL: https://doi.org/10.1145/1290168.1290176.

[246] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. "Long term study of peer behavior in the KAD DHT". In: *Transactions on Networking* 17.5 (2009), pp. 1371–1384. URL: http://doi.acm.org/10.1145/1665838.1665840.

[247] Ralf Steinmetz and Klaus Wehrle. *Peer-to-peer systems and applications*. Springer, 2005.

[248] Daniel Stutzbach and Reza Rejaie. "Capturing Accurate Snapshots of the Gnutella Network". In: *Proceedings of the Conference on Computer Communications (INFOCOM)*. IEEE, 2006. URL: https://doi.org/10.1109/INFOCOM.2006.347.

[249] Daniel Stutzbach and Reza Rejaie. "Evaluating the Accuracy of Captured Snapshots by Peer-to-Peer Crawlers". In: *Proceedings of the Workshop on Passive and Active Network Measurement (PAM)*. Springer, 2005, pp. 353–357. URL: https://doi.org/10.1007/978-3-540-31966-5_33.

[250] Daniel Stutzbach, Reza Rejaie, and Subhabrata Sen. "Characterizing unstructured overlay topologies in modern P2P file-sharing systems". In: *IEEE/ACM Trans. Netw.* 16.2 (2008), pp. 267–280. URL: http://doi.acm.org/10.1145/1373990.1373992.

[251] Daniel Stutzbach, Shanyu Zhao, and Reza Rejaie. "Characterizing files in the modern Gnutella network". In: vol. 13. 1. 2007, pp. 35–50. URL: https://doi.org/10.1007/s00530-007-0079-8.

[252] Lana Swartz. "What was Bitcoin, what will it be? The techno-economic imaginaries of a new money technology". In: *Cultural Studies* 32.4 (2018), pp. 623–650.

[253] Antonio Tenorio-Fornés, Viktor Jacynycz, David Llop-Vila, Antonio A. Sánchez-Ruiz, and Samer Hassan. "Towards a Decentralized Process for Scientific Publication and Peer Review using Blockchain and IPFS". In: *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*. ScholarSpace, 2019, pp. 1–10. URL: http://hdl.handle.net/10125/59901.

[254] "Tor: The Second-Generation Onion Router". In: *Proceedings of the USENIX Security Symposium*. USENIX Association, 2004, pp. 303–320. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[255] Muoi Tran, Inho Choi, Gi Jun Moon, Anh V. Vu, and Min Suk Kang. "A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network". In: *Proceedings of Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 894–909. URL: https://doi.org/10.1109/SP40000.2020.00027.

[256] Florian Tschorsch. "Onions in the queue: an integral networking perspective on anonymous communication systems". PhD thesis. Humboldt University of Berlin, Unter den Linden, Germany, 2016. URL: http://edoc.hu-berlin.de/dissertationen/tschorsch-florian-2016-06-06/PDF/tschorsch.pdf.

[257] Florian Tschorsch and Björn Scheuermann. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies". In: *IEEE Commun. Surv. Tutorials* 18.3 (2016), pp. 2084–2123. URL: https://doi.org/10.1109/COMST.2016.2535718.

[258] Alan Mathison Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London mathematical society* 2.1 (1937), pp. 230–265.

[259] European Union. *European Union Blockchain Strategy*. 2020. URL: https://ec.europa.eu/digital-single-market/en/blockchain-technologies.

[260] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. "A survey of DHT security techniques". In: *ACM Comput. Surv.* 43.2 (2011), 8:1–8:49. URL: https://doi.org/10.1145/1883612.1883615.

[261] Shaileshh Bojja Venkatakrishnan, Giulia C. Fanti, and Pramod Viswanath. "Dandelion: Redesigning the Bitcoin Network for Anonymity". In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.1 (2017), 22:1–22:34. URL: https://doi.org/10.1145/3084459.

[262] Dan S. Wallach. "A Survey of Peer-to-Peer Security Issues". In: *Proceedings of the Symposium on Software Security – Theories and Systems (ISSS)*. Springer, 2002, pp. 42–57. URL: https://doi.org/10.1007/3-540-36532-X_4.

[263] Liang Wang and Jussi Kangasharju. "Measuring large-scale distributed systems: case of BitTorrent Mainline DHT". In: *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*. 2013, pp. 1–10. URL: https://doi.org/10.1109/P2P.2013.6688697.

[264] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. "Attacking the kad network - real world evaluation and high fidelity simulation using DVN". In: *Security and Communication Networks* 6.12 (2013), pp. 1556–1575. URL: https://doi.org/10.1002/sec.172.

[265] Roger Wattenhofer. *The Science of the Blockchain*. 1st. CreateSpace Independent Publishing Platform, 2016.

[266] Duncan J Watts and Steven H Strogatz. "Collective dynamics of small-worldnetworks". In: *Nature* 393.6684 (1998), pp. 440–442.

[267] Joseph Weizenbaum. *Computer power and human reason: From judgment to calculation*. WH Freeman & Co, 1976.

[268] Sam M. Werner, Daniel Pérez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. "SoK: Decentralized Finance (DeFi)". In: *CoRR* abs/2101.08778 (2021). arXiv: 2101.08778. URL: https://arxiv.org/abs/2101.08778.

[269] John Williamson and Theodore H Moran. *Exchange rate regimes for emerging markets: reviving the intermediate option*. Peterson Institute for International Economics, 2000.

[270] Joon Ian Wong. *CryptoKitties is causing ethereum network congestion*. 2017. URL: https://qz.com/1145833/cryptokitties-is-causing-ethereum-network-congestion/.

[271] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* (2014), pp. 1–32. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[272] Karl Wüst and Arthur Gervais. "Do you Need a Blockchain?" In: *Proceedings of Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 45–54. URL: https://doi.org/10.1109/CVCBT.2018.00011.

[273] Guangquan Xu, Bingjiang Guo, Chunhua Su, Xi Zheng, Kaitai Liang, Duncan S. Wong, and Hao Wang. "Am I eclipsed? A smart detector of eclipse attacks for Ethereum". In: *Comput. Secur.* 88 (2020). URL: https://doi.org/10.1016/j.cose.2019.101604.

[274] Beverly Yang and Hector Garcia-Molina. "Designing a Super-Peer Network". In: *Proceedings of the Conference on Data Engineering (ICDE)*. IEEE, 2003, pp. 49–60. URL: https://doi.org/10.1109/ICDE.2003.1260781.

[275] Renlord Yang, Toby Murray, Paul Rimba, and Udaya Parampalli. "Empirically Analyzing Ethereum's Gas Mechanism". In: *Proceedings of the European Symposium on Security and Privacy (EuroS&P) Workshops*. IEEE, 2019, pp. 310–319. URL: https://doi.org/10.1109/EuroSPW.2019.00041.

[276] David Yermack. "Is Bitcoin a real currency? An economic appraisal". In: *Handbook of digital currency*. Elsevier, 2015, pp. 31–44.

[277] Jie Yu and Zhoujun Li. "Active Measurement of Routing Table in Kad". In: *Proceedings of the Consumer Communications and Networking Conference (CCNC)*. IEEE, 2009, pp. 1–5. URL: https://doi.org/10.1109/CCNC.2009.4784962.

[278] Bassam Zantout, Ramzi Haraty, et al. "I2P data communication system". In: *Proceedings of the International Conference on Networks (ICN)*. XPS, 2011, pp. 401–409.

[279] Dirk A Zetzsche, Ross P Buckley, Douglas W Arner, and Linus Föhr. "The ICO Gold Rush: It's a scam, it's a bubble, it's a super challenge for regulators". In: *University of Luxembourg Law Working Paper* 11 (2017), pp. 17–83.

[280] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. "Town Crier: An Authenticated Data Feed for Smart Contracts". In: *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 270–282. URL: https://doi.org/10.1145/2976749.2978326.

[281] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V. Le, and Arthur Gervais. "High-Frequency Trading on Decentralized On-Chain Exchanges". In: *CoRR* abs/2009.14021 (2020). arXiv: 2009.14021. URL: https://arxiv.org/abs/2009.14021.

[282] Shunfan Zhou, Zhemin Yang, Jie Xiang, Yinzhi Cao, Min Yang, and Yuan Zhang. "An Ever-evolving Game: Evaluation of Real-world Attacks and Defenses in Ethereum Ecosystem". In: *Proceedings of the USENIX Security Symposium*. USENIX Association, 2020, pp. 2793–2810. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/zhou-shunfan.

SELBSTÄNDIGKEITSERKLÄRUNG

Ich erkläre, dass ich die Dissertation selbständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42/2018 am 11.07.2018 angegebenen Hilfsmittel angefertigt habe.

Sebastian Henningsen, 15. März 2021