This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

Network Layer Aspects of Permissionless Blockchains

Till Neudecker and Hannes Hartenstein
Institute of Telematics
Karlsruhe Institute of Technology, Germany
Email: {till.neudecker, hannes.hartenstein}@kit.edu

***Abstract*—Permissionless blockchains reach decentralized consensus without requiring pre-established identities or trusted third parties, thus enabling applications such as cryptocurrencies and smart contracts. Consensus is agreed on data that is generated by the application and transmitted by the system's (peer-to-peer) network layer. While many attacks on the network layer were discussed so far, there is no systematic approach that brings together known attacks, the requirements, and the design space of the network layer. In this paper, we survey attacks on the network layer of permissionless blockchains, and derive five requirements: performance, low cost of participation, anonymity, DoS resistance, and topology hiding. Furthermore, we survey the design space of the network layer and qualitatively show the effect of each design decisions on the fulfillment of the requirements. Finally, we pick two aspects of the design space, in-band peer discovery and relay delay, and demonstrate possible directions of future research by quantitatively analyzing and optimizing simplified scenarios.**

We show that while most design decisions imply certain tradeoffs, there is a lack of models that analyze and formalize these tradeoffs. Such models could aid the design of the network layer of permissionless blockchains. One reason for the lack of models is the deliberately limited observability of deployed blockchains. We emphasize that simulation based approaches cope with these limitations and are suited for the analysis of the network layer of permissionless blockchains.

I. Introduction

Permissionless blockchain based systems such as Bitcoin [57], Ethereum [80], and Monero have received widespread attention from the scientific community primarily because of their function as a decentralized consensus system, which can be used to implement, e.g., currency systems or smart contracts. These applications impose certain requirements regarding performance and security on the underlying consensus system. Much effort has been put into a better understanding of the properties of blockchain systems' consensus layer (e.g., [36]). As the consensus layer relies on data transmitted by the network layer, vulnerabilities in the network layer also affect the consensus and application layer. Hence, the security of the overall system also depends on the security of the network layer. Therefore, a precise understanding of the properties of the network layer of blockchains is required.

So far, the network layer of blockchains has been analyzed only with regard to specific attacks such as network based deanonymization attacks (e.g., [10], [31]), double spend attacks (e.g., [46]), or eclipsing attacks (e.g., [4], [41]), or in an empirical way (e.g., [19], [23]). While these works help in the design of the network layer by pointing out certain weaknesses and propose countermeasures against them, they do not show the complete design space of the network layer and do not point out the inherent tradeoffs for some design decisions. This coincides with the observation that many design choices made in the network layer of client implementations seem ad-hoc and without sufficient reasoning. This does not make them bad choices, however, knowledge of which choices are possible and knowledge of the inherent tradeoffs between design choices could result in better choices or justified design decisions.

This paper aims at supporting the design of network layer of permissionless blockchains by reviewing relevant attacks, by giving a comprehensive survey of the design space, and by showing which tradeoffs are implied by design decisions. Furthermore, we demonstrate future research challenges at the example of two aspects of the design space, i.e., the in-band peer discovery mechanism and the randomization of the message relaying. We limit our analysis on the network layer of permissionless[1] blockchains, i.e., neither the underlying network stack (e.g., TCP, IP, Ethernet, 5G [5]), nor the consensus or application layer will be analyzed.

The network layer of blockchain systems is related to two classes of systems that have been analyzed in the past: unstructured peer-to-peer (P2P) networks and anonymity providing networks. Permissionless blockchain systems rely on a public P2P network that is used for information dissemination between participating peers. For a correct functioning of the consensus layer all peers require knowledge of the set of information consensus is to be agreed on (e.g., blocks and transactions). Flooding or gossip protocols are used for the propagation of the required information to all peers of the network. This makes the network layer of permissionless blockchains unstructured P2P networks, which have been used for decades (e.g., Gnutella [68]) and were extensively analyzed (e.g., [44], [52]), however, mostly from a performance perspective or with adversary models not matching the threat to blockchain systems.

Although anonymity providing networks (e.g., Tor [22]) have different requirements regarding information propagation

[1]Permissioned blockchains require pre-established identities of participants, which significantly changes the design space and suggests more centralized architectures. Hence, permissioned blockchains are also not covered by this work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

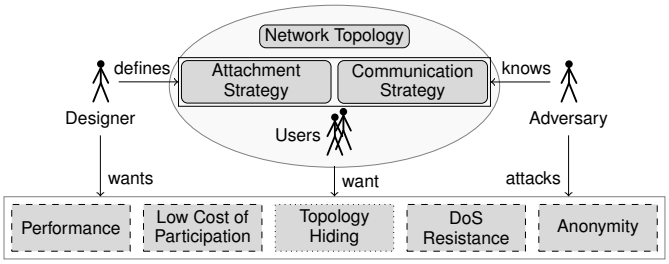

Fig. 1. The network layer of a blockchain system is characterized by the P2P network topology, the client's attachment and communication strategies, and the users' behaviors. The requirements include performance, low cost of participation, anonymity and DoS resistance, and the intermediate goal network topology hiding.

than blockchain based systems, they are similar in the considered adversary models and security requirements. Commonly considered requirements in anonymity providing networks are high performance, low bandwidth cost, resistance to traffic analysis, and resistance to catastrophic denial of service (DoS) [8], which we will use as a basis for our analysis.

Fig. 1 gives a high level overview of the aspects, requirements and actors affecting the network layer of permissionless blockchain systems. The system's P2P network is characterized by its network topology and the behavior of its peers, determined by the client software behavior (i.e., its communication and attachment strategy) and the user behavior. In addition to the (non-security) requirements *performance* and *low cost of participation* and the security requirements *anonymity* and *DoS resistance*, we also consider *network topology hiding* as an intermediate security requirement, because many attacks rely on knowledge of the network topology.

The remainder of the paper is structured as follows. After briefly covering related work in Section II and giving a short introduction into the fundamentals of permissionless blockchains in Section III, we describe requirements and survey known attacks on the network layer of blockchains in Section IV. In Section V we extensively survey the design space of the network layer of blockchains. This survey points out design aspects and their effects on the fulfillment of requirements and enables us to detail on two aspects of the design space (*peer discovery* and *randomized message relaying*) and present approaches for their quantitative analysis and optimization in simplified scenarios in Sections VI and VII. Finally, Section VIII summarizes the insights and highlights future research directions.

II. Related Work

We will briefly discuss related work that covers network security in blockchain systems or P2P networks on an abstract level. Discussion of related works with more focus (e.g., specific attacks) is given in Sections IV-C and V.

Gervais et al. [36] give a thorough security analysis of proof-of-work blockchain systems, however, their focus is on the consensus layer (i.e., block generation) whereas the network layer is abstracted. Troncoso et al. [75] show a broader perspective covering numerous systems apart from Bitcoin and Tor but also abstract from the network layer. A recent paper by Delgado-Seguara et al. [20] explores the characteristics of the peer-to-peer network established by Bitcoin, but abstracts from the design space of the network layer. Lua et al. [52] focus on file sharing peer to peer networks and leave out anonymity goals and strong adversary models. There are several surveys covering Bitcoin in general [12], [76], the security of Bitcoin [16], and the privacy and anonymity of permissionless blockchains [48].

III. Fundamentals: Permissionless Blockchains

We will now briefly sketch the fundamentals of permissionless blockchains. A comprehensive introduction to the topic can be found in, e.g., [58] and [76]. Permissionless blockchains are P2P networks that maintain a state and allow modification of that state by users of the system. Users formulate changes to the state as *transactions*, which are published on the network. For instance, a transaction in Bitcoin moves funds between entities of the system. In order to ensure that only authorized users change the state (e.g., only the owner of funds should be able to transfer them), transactions are usually cryptographically signed. The verification of the signature can be done using previously stored state (e.g., the previous reception of funds specifies the public key of the new owner of the funds).

In an ongoing process a set of new transactions is accumulated into one *block*, which is also published on the P2P network. Each generated block contains the hash value of the previous block. This *mining* process creates a chain of blocks, referred to as *blockchain*. A blockchain has the property that any change to an old block (e.g., modifying or removing a transaction) changes the hash values of all subsequent blocks. Therefore, blockchains are often regarded as *immutable* (assuming authenticity of the current block hash).[2]

The mining process needs to ensure that the maintained state does not become inconsistent, e.g., by having multiple chains of blocks that contradict each other. This requires that the creation of blocks is regulated by a consensus mechanism. In contrast to permissioned blockchains, the set of participants allowed to create blocks is not known beforehand in permissionless blockchains. This makes permissionless blockchains vulnerable to Sybil attacks [24] and prohibits the use of byzantine fault tolerance consensus protocols such as PBFT [14]. Instead, permissionless blockchains usually employ a proof-of-work mechanism [7], [26], i.e., the creation of valid blocks requires solving a computationally expensive hash puzzle, which limits the ability to contribute to the consensus process to the computational resources of the participant. The high energy consumption of proof-of-work mining [18] gives also rise to discussions on sustainability [81].

[2]Strictly speaking, a blockchain does not guarantee immutability because peers can modify data on the blockchain. However, such modification becomes evident to other users. Hence, the term *tamper-evident* is also used to describe blockchains.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

If two blocks are mined at the same time, the blockchain *forks*, i.e., there are two chains of equal length, which differ in the newest block. This implies that the system state is (temporarily) not consistent. However, the probability of two chains growing synchronously decreases exponentially over time. Therefore, one chain will eventually become the longest chain. Because only the longest chain holds the system state, the state of the shorter chain is ignored and consistency is achieved.[3]

IV. System Requirements

Based on the sketch given in Fig. 1 we will now characterize the network layer of permissionless blockchain systems. Bitcoin [57] serves as a prototype for the considered scenario. However, we emphasize that our definition also matches most other permissionless blockchain systems and parts of this work are also applicable to a wider range of unstructured P2P networks.

A. Functional Requirements

In contrast to a private blockchain or a permissioned blockchain, in a permissionless blockchain there is neither a restriction on the ability to read from the blockchain (this ensures public verifiability) nor a requirement for pre-established identities for write access to the blockchain [83]. In order for anyone to join the system, there need to be enough peers on the network that accept incoming connections. There may be peers that are not reachable (e.g., because they are behind a NAT), however, the **openness** of the system can only be guaranteed with a sufficiently large share of the reachable peers. Although unreachable peers can only connect to reachable peers, they can still serve the system by increasing the network's robustness (cf. Section V).

In order to provide public verifiability and allow peers to create blocks, all relevant data must be accessible by peers. Therefore, the main requirement of the network layer of blockchains is the **dissemination of information** among all participants. Peers need to be able to retrieve historic information from the network (e.g., when a new peer initially joins the network), but also need to stay informed continuously about new information. To ensure a fast dissemination of new information, a flooding or gossip mechanism is used, that broadcasts new information to all peers. There may be clients that are unable to process that amount of data (e.g., because they are running on a mobile device), which only get a subset of data relayed (e.g., SPV clients in Bitcoin [40]). These clients, however, put some trust in the peers they connect to and assume that these peers do not withhold messages.

In contrast to (communication) anonymity providing systems, which ensure confidentiality of the exchanged data between a certain sender-receiver pair, blockchain systems publish information to all participants. Therefore, the data is not encrypted, which makes each piece of information (e.g., a transaction) uniquely identifiable and distinguishable. Encryption of the connections between peers was proposed [72], however, without pre-established identities of peers, this approach is vulnerable to man in the middle attacks as authenticity cannot be protected. Authentication of peers, which requires the out of band exchange of a public key, has also been proposed for Bitcoin [4], [71], and is implemented in Ethereum and Tor.

B. Non-Functional Requirements

Back et al. [8] identified two non-functional requirements for anonymity providing networks: performance and low bandwidth cost. Although the network layer of blockchains and anonymity providing systems differ in many ways, both requirements still apply. We generalize the goal of a low bandwidth cost and require a low cost of participation, which implies low bandwidth costs.

Performance: Information dissemination should be fast. For a blockchain system, a slow dissemination of information implies a longer time until consistency on the information is reached. This can facilitate attacks on the application layer such as double-spends (cf. Section IV-C). Furthermore, the efficiency of proof-of-work blockchains depends on fast information dissemination [36]. A metric that quantifies the performance is the delay between the initial sending of a piece of information until the time that n percent of peers have received that information.

Low Cost of Participation: As permissionless blockchains aim to be open for participants to join the network, participants wishing to run a peer on the network should not be faced with unbearable costs.[4] The costs of running a peer include the required bandwidth, computation, and storage costs. Reducing these costs enables more users to run a peer, which improves the overall reliance of the system. On the other hand, a large number of peers make the consideration of scalability issues necessary. Metrics that quantify the cost of participation are the number of bytes a peer has to send and receive depending on the application layer load (e.g., the number of transactions) during a time period, or the required storage.

C. Security Requirements - Attack Survey

In order to understand the security requirements we will first revisit network based attacks on Bitcoin, visualized in Fig. 2 using the concept of attack trees [70] (with OR nodes only).

One possible goal of an adversary is the deanonymization of users by associating network layer information (e.g., IP addresses) to application layer information (e.g., transactions). Several works show that this is possible by exploiting, e.g., Bloom filters used for SPV clients, tunneling protocols or the general message relay behavior. The gathered information could be further used to identify money flows

[3]The consistency model of permissionless blockchains is subject to discussion and is usually referred to as eventual consistency [79], or as a probabilistic form of strong consistency [73].

[4]While users can participate in the blockchain without running a peer by delegating the communication with the network to trusted parties, we only consider the direct communication with the network by running a peer as participation from a network-level perspective.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

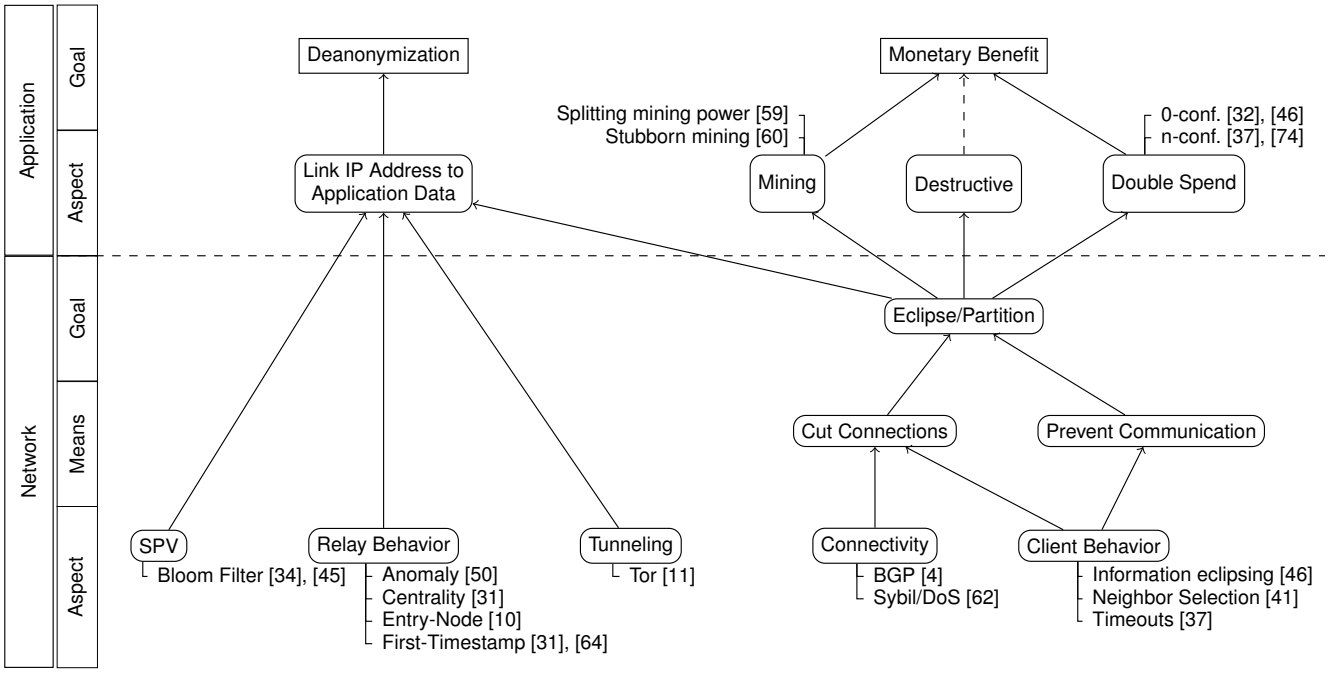

Fig. 2. Known network-based attacks on permissionless blockchains at the example of Bitcoin, visualized as attack trees.

Another goal of an adversary could be to gain a monetary advantage by either earning disproportionally high mining rewards, by double spending funds, or by performing a destructive Goldfinger attack[5]. Destructive attacks can also be ideologically or politically motivated without any monetary incentive. All published attacks have in common that some form of interruption of information flow between peers on the network is required. This can be either the eclipsing of single peers from certain information or the complete partitioning of large parts of the network. Two general methods to achieve this were proposed: Either to directly attack the connections between peers, i.e., after a successful attack the victim peer has no connections to other honest peers, or to prevent communication from and to victim peers, i.e., after a successful attack the victim peer has functioning connections to other peers but does not receive the required information over them. Preventing communication can be achieved by exploiting client behavior. Cutting connections can also be achieved by exploiting client behavior, but also by directly attacking the underlying network stack (e.g., TCP, IP).

Based on the discussed attack tree we can identify two security requirements for the network layer: Prevent linkage of IP addresses to application layer information (**anonymity**) and prevent interruption of the information flow by eclipsing (**DoS resistance**).

A number of attacks (e.g., [10], [31], [62]) require the adversary to know the network topology of the P2P overlay network. As this information is not publicly known and known to be hard to infer [63], an intermediate goal for an adversary could be to approximate the topology of the underlying P2P network. Although many attacks are still possible without knowledge of the network topology, lack of such knowledge causes a less precise attack which generally requires more resources to be spent by the adversary. Hence, we also consider **topology hiding** an intermediate security requirement.

Metrics that quantify topology hiding are the precision and recall with which a certain adversary can estimate connections of the network topology. These metrics will be used in Section VI. Metrics that quantify anonymity are the precision and recall with which a certain adversary can link network layer to application layer information. A metric that quantifies DoS resistance is the required amount of resources required to execute a DoS attack for a certain adversary.

Fig. 2 does not include attacks without interaction with the network layer, e.g., selfish mining [28] or address clustering [67]. For a comprehensive survey of such attacks see [16].

D. Adversary Models

The attacks discussed in the previous section made various assumptions on the ability of the adversary. Therefore, also the discussed metrics of the security requirements depend on the adversary model. We will now discuss several aspects of adversary models.

For network-based attacks, the most critical aspect is the **network power** of the adversary. An adversary can easily run a peer that connects to all reachable peers in the network.[6] Such *monitoring peers* have been previously shown to be able to connect to several thousand peers on the Bitcoin

[5]In a Goldfinger attack the adversary earns money by destruction of the system, e.g., by going short on the cryptocurrency, or by blackmailing [51].

[6]The number of reachable peers on the Bitcoin network varied between 6,000 and 12,000 in 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

P2P network [63] using standard hardware and requiring a bandwidth of less than 100Mbit/s. Adversaries with more resources could perform a *Sybil attack* on the network by inserting a large number of peers to the network. As the cost of running a network peer should be low (cf. Section IV-B), the number of Sybil peers on the network could outnumber the number of honest network peers. The adversary could run the Sybil peers on own hardware, use cloud service providers, or use botnet services. Finally, an adversary with *access to core internet infrastructure* (e.g., ISPs, internet exchange points, intelligence services) is additionally able to monitor and manipulate traffic, e.g., by hijacking BGP routes [4] or simply by dropping packets.

The **computation power** available to an adversary can be either modeled to allow the adversary to only run the peers required for the attack, or the adversary can also have a share of *mining power* under control. With mining power available to the adversary, network based attacks that result in disproportional mining rewards are possible (e.g., [60]).

An additional variable in the adversary model is the **attack duration**. Thanks to cloud services, it may be cheap for an adversary to spawn several thousand Sybil peers for a short period of time [61]. However, carrying out such an attack for a longer period increases the adversary's costs.

Network power, computation power, and attack duration are highly specific to each single adversary. In contrast, there are similarities among all adversaries: Adversaries in permissionless blockchain systems are always able to create new pieces of information (e.g., transactions) and insert them to the network. Furthermore, all adversaries are aware of the complete source code of all client's software running on the network. Client software is usually published open source in order to establish trust in the implementation and the system. Although it is possible for single parties to use a modified client with a deviating behavior, for the majority of users this is not viable. Hence, adversary models should assume that it is not possible to hide any behavior from the adversary (cf. Kerckhoffs' principle [47]).[7]

E. Related Requirements and Adversary Models

We will now discuss additional requirements and adversary models that have been considered for permissionless blockchains or related systems.

1) Requirements: Scalability has been identified as a separate non-functional requirement of anonymous communication systems [38]. We will treat scalability as the dependance of the overall *cost of participation* and the *performance* on the number of peers.

A high level of decentralization, i.e. the absence of one or a few authorities that control a large share of the system components (e.g., network peers, mining power), is also a requirement of permissionless blockchains [35]. From a network layer perspective, a low cost of participation is a requirement for decentralization. Centralization in a system can be modeled by considering adversary models that have, e.g., a large share of mining power or a large number of (Sybil) peers. Therefore, while decentralization is an important requirement, especially on the consensus layer, we do not treat decentralization as a distinct requirement.

Finally, there can be a lack of incentives to actively participate in the P2P network and actually forward transactions and blocks to other peers [6], [27]. For instance, miners can increase their revenue by withholding transactions with high transaction fees from other miners. Hence, incentive compatibility can also be regarded as a requirement. While we acknowledge the importance of incentive compatibility, the analysis of incentive compatibility requires the common consideration of network, consensus, and application layer aspects using game theoretic approaches and is therefore out of scope in this work.

2) Adversary Models: Slightly different adversary models than the ones discussed before have been used in the field of rumor source detection (e.g., [30]). In this setting the only goal of the adversary is the identification of the source of a rumor that is propagated across the network. This equals the attack of anonymity by linking network data (*the rumor source*) to application data (*the rumor*). The adversary is either modeled to obtain a *snapshot* of the propagation of a rumor at a certain point in time or to have a number of *spy nodes* in the network that gather the time of reception of a rumor as well as all possibly attached metadata.

One main difference in the considered adversary models is that the adversary in rumor source detection is usually assumed to know the topology of the network. Although this assumption can be valid in certain scenarios, it does not hold in the case of a peer-to-peer network with dynamically established links.

V. Design Space Survey

We will now analyze the design space of the network layer of permissionless blockchains from the perspective of a developer, who maintains the source code of a client software. As the network is assumed to be open, users are still free to use modified client software with deviant behavior. Table I summarizes the affected requirements and further readings for each aspect that will be discussed. The second column shows the requirements that are mainly affected by a certain design aspect. Tradeoffs between two or more requirements are marked with the symbol $\leftrightarrow$.

There are two strategies that can be modified in the client that affect the network layer of the system: First, the **attachment strategy** defines which connections to other peers are established. Second, the **communication strategy** defines how clients communicate with their neighbors.

As all public P2P networks require interconnection between its peers, the attachment strategies in different P2P networks are very similar among a wide range of P2P networks including permissionless blockchains. This allows the analysis of the attachment strategy of blockchain systems along with the attachment strategy of systems like Tor in order to

[7]Of course the client can make use of pseudorandom number generators, the output of which remains unknown to the adversary.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

TABLE I
DESIGN SPACE OF THE NETWORK LAYER OF PERMISSIONLESS BLOCKCHAINS

	Main Affected Requirements & Tradeoffs	Further Readings
Peer Discovery	Functional requirement: establish and maintain list of IP addresses of other peers.	
Out-of-Band	DoS Resistance, Anonymity, Topology Hiding	Remarks: required for bootstrapping, trusted third party
In-Band	[Performance, DoS Resistance] ↔ [Topology Hiding]	Section VI, [56], [41]
Neighbor Selection	Functional requirement: establish and maintain connections to network peers	
Information Sources	—	Remarks: IP address based, own observation, reputation
Number of Connections	[Performance, DoS Resistance] ↔ [Cost of Participation]	[1], [2]
Incoming Connections	[Performance, DoS Resistance] ↔ [Cost of Participation]	[3], [9], [10], [41]
Topology Generation	[Performance, Cost of Participation] ↔ [DoS Resistance]	[29], remarks: requires trusted information
Stability	[DoS Resistance] ↔ [Topology Hiding]	[15]
Connection Anomaly Detection	DoS Resistance	[4]
Communication	Functional requirement: information propagation	
Information Sources	—	[77], remarks: Message content, meta-information , side-channel
Push vs. Announce-and-Request	[Performance] ↔ [Cost of Participation]	[37]
Flooding vs. Gossip	[Performance, DoS Resistance] ↔ [Cost of Participation]	[66]
Relay Delay	[Performance] ↔ [Anonymity, Topology Hiding]	Section VII
Message Accumulation	Performance, Cost of Participation	Remarks: effect on topology hiding and anonymity unclear

identify similarities and varying approaches. In contrast, the communication strategy is heavily driven by application layer requirements and makes a comparison between blockchain based systems and systems used for applications such as file sharing not beneficial.

We will now analyze the design space by covering all aspects of the attachment and communication strategies, which either appear in deployed systems, or were proposed or discussed. Although these aspects characterize the network layer of known systems, new systems may include aspects that are not covered in our analysis.

A. Attachment Strategy

The attachment strategy defines how clients establish connections to remote peers. In order to establish outgoing connections, a client needs to discover the IP addresses of other peers (*peer discovery*). Then, the client has to decide to which peers it establishes connections and how it handles incoming connections (*neighbor selection*).

1) Peer Discovery: The main goal of peer discovery is to establish and maintain a set of reachable IP addresses of other peers in order to establish connections to them. Peer discovery can be done using out-of-band communication with one or more *seed nodes*) that provide IP addresses of reachable peers. If a client is connected to at least one peer on the network, peer discovery can also be performed in-band by requesting IP addresses of reachable peers from neighbors.

Out-of-band peer discovery with one or more seed nodes (e.g., DNS server, IP addresses hard coded to the client) is in general required for bootstrapping.[8] Although a large number of seed nodes can be used, it is still some form of centralization that can affect the requirements DoS resistance, anonymity, and topology hiding.

Discussion: Malicious seed nodes might return only IP addresses of peers under its own control, hence enabling eclipsing attacks on peers that rely solely on information from that seed.[9] Furthermore, malicious seed nodes could try to attack anonymity by linking IP addresses of requesters to application layer data that is later transmitted via the peer returned in the seed's reply. Finally, malicious seed node operators could try to infer connections by IP addresses of requesters to IP addresses returned by the seed node. Adversaries might also try to perform DoS attacks on the seed nodes itself, making it impossible for peers to connect to the network.

In order to prevent these attacks, a large number of seed nodes operated by different parties is required. Clients should request IP addresses from multiple seed nodes operated by different parties to minimize chances that all seed nodes are compromised. Connections should then be established to IP addresses received from different parties. To improve topology hiding, clients should receive a large number of IP addresses but only connect to a small subset. Obviously, all measures cause a bandwidth overhead for clients and seed node operators, thus increasing the cost of participation.

Examples: Tor uses out-of-band peer discovery with a hard coded list of directory authorities along with their public keys[10]. The Bitcoin client[11] and the Monero client have hard coded lists of DNS seeds for bootstrapping.[12] The communication to the DNS seeds is not authenticated. Bitcoin has an operator policy that requests from seed nodes to provide

[8]Another possibility is random address probing [21], where a clients tries to connect to peers randomly selected from the IP address space. Random address probing comes with significant drawbacks, especially a high bandwidth cost, bad performance, and it is practically infeasible in IPv6 address space.

[9]The IP addresses received using in-band communication from that peer are then also controlled by the adversary.

[10]https://www.torproject.org/docs/faq#KeyManagement and https://github.com/torproject/tor/blob/f9615f9d770e035829e1ff238980d6ac6e852150/src/or/config.c#L992

[11]When discussing client implementation aspects, we use the term *Bitcoin* to refer to the reference client bitcoind (https://github.com/bitcoin/bitcoin).

[12]Bitcoin: https://github.com/bitcoin/bitcoin/blob/13f53b750dc09cb59192b2aa4ac8e499ee36e1ca/src/chainparams.cpp#L127, Monero: https://github.com/monero-project/monero/blob/75563db6e36f044ca0fd08722e2b29a3c950430a/src/p2p/net_node.h#L133

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

unbiased samples of functioning network peers and to not use data gathered from operating the seed node to attack the anonymity of users.[13]

Ethereum has a list of IP addresses along with the public keys of the peers hard coded into the client.[14] Other methods for bootstrapping include IRC channels (e.g., used by Gnutella and formerly used by Bitcoin).

In-band peer discovery can be used once at least one connection to another peer is established. Clients can either request IP addresses from their neighbors or clients can send IP addresses unsolicited to their neighbors. The requirements performance and DoS resistance are affected by the peer discovery, because it determines the speed of establishing connections. The requirement topology hiding is also affected, because in-band peer discovery can be used to infer connections between peers.

Discussion: The announced IP addresses should be reachable with substantial probability, i.e., a successful connection to the announced address should have been made in the past. However, the announced IP addresses should not indicate the connections of a peer. If a client naively announced its neighbor's IP addresses as reachable peers, adversaries could easily infer the connections of that client. However, as peers join and leave the network, it is important to distinguish between peers that are still online and peers that already went offline. Otherwise, old IP addresses keep being announced on the network because they were reachable in the past. A detailed analysis of these tradeoffs is presented in Section VI.

Examples: Bitcoin clients can request IP addresses from their neighbors by sending GETADDR messages. Furthermore, clients can send IP addresses unsolicited to their neighbors. Peer discovery has been successfully used to infer the network topology of the Bitcoin P2P network [56]. Ethereum uses a Kademlia-like [54] system to discover IP addresses of peers based on their public key.[15] This mechanism was vulnerable to eclipse attacks [53].

2) Neighbor Selection: The main goal of neighbor selection is the establishment of connections to other peers so that the requirements of the system (e.g., performance, DoS resistance) are satisfied. Generally speaking, a peer can choose to which peers it establishes outgoing connections and from which peers it accepts incoming connections. For this decision, the following questions need to be answered:

- Which information sources are available for the assessment of remote peers?
- How many outbound connections are established? How many inbound connections are accepted?
- Does the selection of neighbors aim at creating a certain network topology?
- Are connections maintained for as long as possible?
- Is the correct functioning of connections monitored?

[13]Bitcoin DNS seed operator policy: https://github.com/bitcoin/bitcoin/blob/57b34599b2deb179ff1bd97ffeab91ec9f904d85/doc/dnsseed-policy.md

[14]https://github.com/ethereum/go-ethereum/blob/79b11121a7e4beef0d0297894289200b9842c36c/params/bootnodes.go

[15]https://github.com/ethereum/devp2p/blob/master/rlpx.md#node-discovery

Information Sources: Once a client has a set of possibly reachable IP addresses obtained through peer discovery, the client can either randomly select peers to connect to from that set, or discriminate remote peers based on information about that peer. Discriminating remote peers can be advantageous in order to (1) prevent adversaries from monopolizing all connections of a client (i.e., improving DoS resistance), or (2) enhance the performance or reduce bandwidth cost by creating certain network topologies.

Discussion: A peer can utilize three types of information about foreign peers: (1) only the IP address and static information associated with it, (2) information based on own observations in the past, (3) information provided by others. The IP address of a peer can be used to identify, e.g., from which IP address ranges, autonomous systems, or geographic regions other peers come from. It is known to the client even before a connection is established. On the other hand, using information based on own observations requires the previous establishment of connections to that peer.[16] Using information provided by other peers on the network (i.e., reputation) is vulnerable to Sybil attacks [24]. Hence, one or more trusted entities providing that information are required.

Examples: IP address information is used in Bitcoin to limit the number of outgoing connections per IP address range in order to improve DoS resistance by preventing the client from establishing too many connections to adversaries with limited IP address resources. Furthermore, taking AS level information into account has been proposed for Bitcoin [4] and Tor [42]. Information based on own observations is used in Bitcoin to blacklist IP addresses that were misbehaving in the past for a certain amount of time. Tor uses trusted directory authorities to provide information such as available bandwidth and availability (i.e., uptime) to clients in order to mitigate Sybil attacks with only a short duration and also to enhance performance by providing information required for load balancing.

Number of Connections: The most basic parameter of the attachment strategy is the number of connections a peer establishes, both, inbound and outbound. Typically, a client establishes a certain number of outgoing connections and may allow up to a certain number of incoming connections. The number of connections is mainly a tradeoff between the cost of participation on the one hand and performance and DoS-Resistance on the other hand.

Discussion: In a flooding network, the bandwidth cost of a peer increases in the number of connections. When naively flooding messages, the increase is linear in the number of connections as every message will be sent over every link once. However, using a two-legged process for flooding (cf. Section V-B), where new messages are first announced and only sent on request, can reduce the overall bandwidth cost to be sublinear in the number of connections. To which peers

[16]Without any identification mechanism of peers, a peer's identity can only be coupled to the IP address it uses. With dynamic IP addresses, this can lead to false association of information and peers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

a client is connected has no significant effect on the total bandwidth cost.

A large number of connections per peer can reduce the propagation delay if the communication strategy makes use of the connections. Many connections can also lead to a reduction in the network diameter, improving propagation speed and enhancing robustness of the network. The required effort for an adversary to isolate a single peer from the network or to partition the whole network increases with the number of connections the clients establish. Many models for the analysis of the effect of the number of connections on performance and security properties of the network have been published (e.g., [1], [2]).

Examples: The default number of outgoing connections for Bitcoin is 8. It has been proposed to increase this number in order to enhance DoS resistance [41].

Incoming Connections are less trustworthy than outgoing connections, because even a very limited adversary (i.e., a small number of Sybil peers) can establish a large number of incoming connections to other peers. The maximum number of incoming connections is, as with the total number of connections, a tradeoff between the cost of participation on the one hand and performance and DoS-Resistance on the other hand. However, DoS resistance increases less with more incoming connections than with the same number of outgoing connections.

Discussion: Allowing a large number of incoming connections enables other peers to establish (more trustworthy) outgoing connections. Furthermore, it is the only possibility for clients that are not able or willing to accept incoming connections to establish connections to the network at all. These peers, although not publicly offering their service, help in connecting the network further, and these peers are hard to identify and attack for adversaries without access to core infrastructure. Hence, allowing incoming connections to be made also improves DoS resistance for the peer itself.

The fact that adversaries can easily establish a large number of incoming connections led to the discussion of several options. In order to prevent information eclipsing, it was proposed to not allow incoming connections when accepting zero confirmation payments in Bitcoin [9]. While certain peers may opt to establish only outgoing connections, the overall network requires peers to accept incoming connections.

A primitive but still possibly effective attack is to use up all incoming connections slot from all peers on the network. This would prevent honest peers from establishing connections. To increase the cost of such attacks it has been proposed to require peers to solve a proof-of-work in order to establish outgoing connections [3], [10], which would also increase the cost of participation. Furthermore, it was proposed to limit incoming connections based on IP address information [41].

Examples: The default number of allowed incoming connections in Bitcoin is 117. The number of unreachable peers in the Bitcoin network (i.e., peers behind NATs or peers not allowing incoming connections) has been estimated to be 155,000 or more in May 2017 [78]. With about 5,500 peers reachable via IPv4 during that time, there are almost 30 unreachable peers per reachable peer, according to the estimate.

Topology Generation: Based on the information sources available to the client, the client can decide to which peers connections are established. This decision affects the resulting network topology, which has a strong effect on the requirements cost of participation, performance, and DoS resistance.

Discussion: Proposals were made to increase performance by favoring the establishing of connections to peers in geographic proximity [29]. Geographic proximity can be easily deducted using IP address information and freely available databases, hence, the required information can be easily obtained. However, although this idea might improve the network's performance, it comes at the cost of highly reduced DoS resistance and robustness against random failure. As the number of long distance links such as inter-continental links is reduced, failure of these connections due to random error or attack can cause the network to partition.[17]

It might also be desirable to create network topologies with certain node degree distributions for performance reasons. For instance, scale free networks result in faster information propagation compared to random Erdős–Rényi (ER) graphs for larger networks [13]. However, there are several issues with these approaches: First, in order to create a certain network topology apart from a random graph, clients need information about the node degree of others (e.g., in order to perform some form of preferential attachment). This information can only be provided by the peers itself (peer may lie about their connection count), or by a trusted entity that is able to monitor connections between peers. Despite their better performance, scale free networks were shown to have a lower resistance to targeted attacks, i.e., to an attack where the adversary knows the network topology and attacks and removes specific peers [2]. As ER graphs exhibit a high attack tolerance it is questionable whether it is a good idea at all to change the network topology to not be a random graph.

Examples: One implemented example of a topology generating attachment strategy is the load balancing in Tor. A network wide load balancing is implemented by choosing the probability to establish a circuit through a certain peer according to its relative bandwidth share of the whole network.

Stability: Once connections are established, clients can either try to keep connections for the longest possible duration (i.e., keep the network topology as static as possible) or can deliberately disconnect from connected peers after some period and connect to other peers. The stability of the network topology affects DoS resistance and topology hiding.

Discussion: On the one hand, a static network topology makes it harder for adversaries to infiltrate the network with Sybil peers, because connections between honest peers remain

[17]The distinction between error and attack tolerance of a network implies that an adversary knows the network topology at least partially and is able to selectively attack connections or peers. In order to enhance the DoS resistance of a network it is advantageous to hide the network's topology from an adversary. An adversary that does not know the network topology has to fall back to randomly attack peers of the network, which equals random failure of peers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

as long as possible and only cease to exist due to churn or DoS attacks. On the other hand, a static network topology makes inferring the network topology easier, which enables DoS attacks on central peers of the network and also facilitates deanonymization attacks that rely on knowledge of the network topology (e.g. [31]).

Examples: For peers without any incoming connections it was proposed to establish new outgoing connections between publishing two transactions in order to avoid deanonymization [10]. A beneficial effect of continuous changes to the network topology has also been suggested against AS hijacking based DoS attacks [4]. A detailed discussion of this aspect is presented in Section VI.

Connection Anomaly Detection: In order to avoid being eclipsed, a peer relies on its neighbors to relay relevant information. Clients can monitor their connections and terminate connections to peers that do not behave as expected. This can improve DoS resistance but can also enable DoS attacks.

Discussion: One scenario in eclipsing attacks is that the adversary tries to monopolize all connections of a peer by inserting Sybil peers that stop relaying messages to the peer at some point in time. In order to counter such attacks a client could monitor the rate at which neighbors relay messages. If that rate is significantly lower than the rate observed in the past (or by other neighbors), the client should establish additional connections to avoid being eclipsed. Obviously, the message rate varies over time so that false positives and false negatives can occur. However, as the cost of temporarily establishing more connections is low, clients at risk of being eclipsed should employ this measure to enhance DoS resistance.

In case a client observes a significantly lower message rate from a certain peer, the client could choose to terminate the connection to that peer. On the one hand this would make monitor and Sybil attacks more expensive as the peers of the adversary are then required to relay messages to their neighbors. On the other hand, such a measure would detain users with minimal resources to passively participate in the system.

Examples: Bitcoin monitors connections in the sense that neighbors which send messages not compliant with the protocol are disconnected and blacklisted. This mechanism has been exploited to disconnect Tor exit nodes from the Bitcoin network [11], i.e., exploiting an anti-DoS mechanism for a DoS attack. Anomaly detection was also proposed to include metrics such as the round-trip time to neighbors in order to detect other kinds of attacks (e.g., attacks on routing) [4].

Although anomaly detection has been proposed in the past, there is a lack of models that actually can be used for monitoring and detection. Such models should reliably predict message rates and provide configuration options for balancing the affected tradeoffs.

B. Communication Strategy

Whenever a client creates a new message (e.g., a transaction or block in Bitcoin), that message needs to be disseminated to all other peers on the network. As the creating client is only connected to a small number of peers, it relies on other peers that relay the message to all other peers. The communication strategy of a client decides at runtime for all messages received in the past, which of these messages are relayed to which neighbors at which point in time and how this relaying is implemented.

For this decision, the following questions need to be answered:

- Which information sources are available?
- Are messages pushed or announced and pulled?
- To which neighbor are messages relayed?
- When are messages relayed?
- Is each message treated separately or are messages aggregated?

Information Sources: The client can either treat each message equally, or adapt the communication strategy according to additional information on the message.

Discussion: Strategies might use the content of the message in order to decide relaying times and peers. In contrast to the message content, which serves the actual application, additional meta information can be sent along with the message to provide information to the communication strategy of other peers. Finally, side-channel information can be any information transmitted via the network or collected locally that can be used by the client.

Examples: One example for using the content of the message is the different treatment of blocks and transactions in Bitcoin, where blocks are immediately relayed whereas random delays are applied before relaying transactions. Another use of the message's content can be to treat own messages (e.g., transactions created by the user of the client) differently from relayed messages. That way, a dissemination strategy that enhances anonymity (at the cost of other goals) can be used for initial sending of a message only. Further relaying of the message can then utilize strategies with better performance.

Dandelion [77] proposes to use a two phase communication strategy and to indicate the current phase via metainformation attached to the message. Clients can then apply the currently selected phase of the communication strategy in order to enhance anonymity.

Meta information could also be used as a general means for users to express their personal tradeoff between anonymity and performance for a certain message by setting a suggested mean message delay value. As the meta information is relayed to a client's neighbors, even for the weakest adversary models the adversary learns of this information. Hence, the use of meta information only makes sense if the advantages in a better communication strategy outweigh the disadvantages of an additional information source for the adversary.

Side-channel information is used when Bitcoin SPV clients tell their neighbors which transactions should be relayed by sending a bloom filter. Because bloom filters can be used to attack the anonymity of users, the design of privacy-preserving bloom filters [45] should be considered. Other proposals to use side-channel information include to send Canary status messages upon detection of double spends [65].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

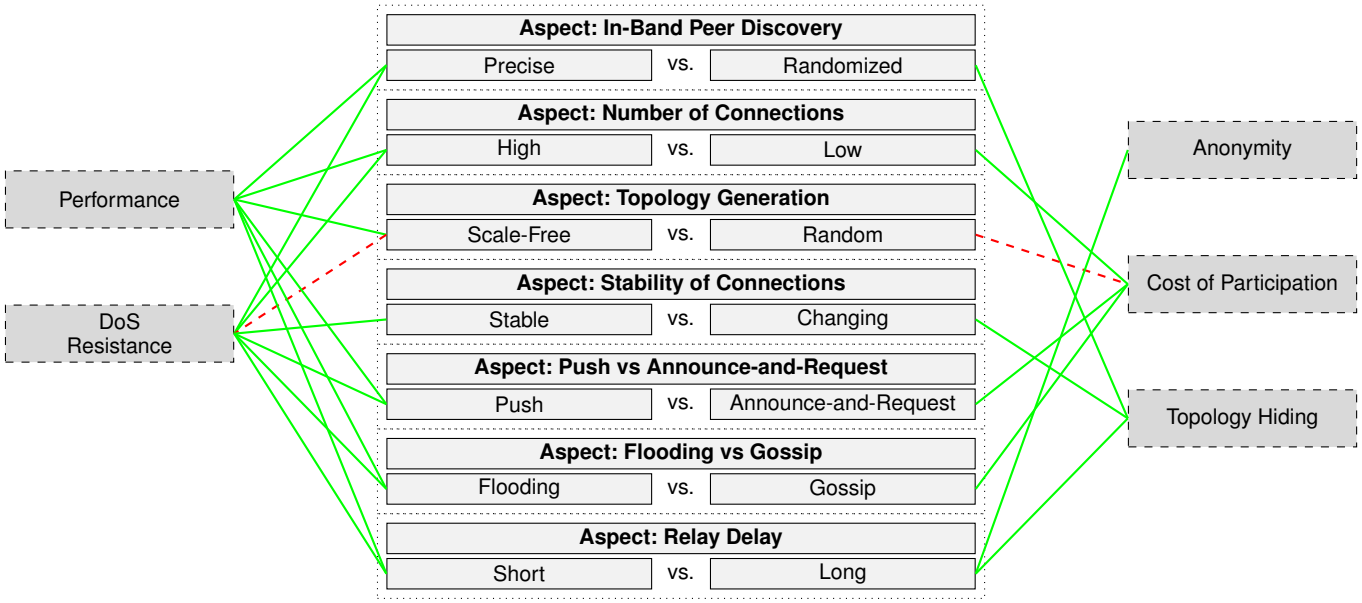

Fig. 3. Exemplary design choices and their effects (solid green = positive, dashed red = negative) on the fulfillment of requirements. For each shown aspect two possible choices are given (e.g., a short and a long relay delay). Redundant effects (e.g., if a short relay delay has a positive effect on performance, a long relay delay obviously has a negative effect on performance) are omitted for the sake of better readability.

Push vs. Announce and Request: A protocol can be designed to directly relay (push) new messages to neighbors, or it can be designed to use a two-legged process where new messages are first announced to neighbors using some form of ID (e.g., the messages hash value). The neighboring client can then check whether the message has already been received and can request the message if necessary (announce-and-request). This aspect is mainly a tradeoff between performance and cost of participation.

Discussion: Push results in a faster propagation than announce-and-request. When pushing a message, only one latency between peers elapses until the message is delivered, with announce-and-request three latencies elapse.

The average bandwidth cost of both approaches can be calculated given the average size of messages (s_m), the size of the ID (s_h), and the probability of request (p_r). The average bandwidth cost per message per connection is s_m for a push strategy, and $s_h + p_r(s_h + s_m)$ for an announce-and-request strategy. The relative bandwidth saving can be calculated as $s_h/s_m(1 + p_r) + p_r$. For instance, assuming $s_h = 32\,\text{Bytes}$, $s_m = 500\,\text{Bytes}$ and $p_r = 1/16$, which is a very rough estimate of the parameters for Bitcoin transactions, announce-and-request only consumes 13 % of the bandwidth of push.

Only in cases where the additional latency is significant, or where the messages are very small compared to their ID and the probability of request is very high (i.e., in sparsely connected networks), a push protocol is favorable. Even in these systems, peers still need to provide a mechanism for other peers to request messages. For instance, new participants in blockchain systems need to access historic data in order to verify new messages.

Examples: When using announce-and-request, clients need to keep track of requested messages and monitor whether the message is in fact delivered by the remote peer. The timeout mechanism used in Bitcoin for that purpose was vulnerable to a DoS attack where an adversary announces new blocks to the victim, but does not deliver the blocks [37]. The block synchronization mechanism in ethereum was also vulnerable to an attack that delayed the reception of valid blocks at remote peers [82]. Proposed countermeasures include using dynamic timeouts, penalizing non-responding peers or requesting the message from multiple peers [37].

For the transmission of blocks in Bitcoin it has been proposed to announce a new block by sending the block header, which is only 80 Bytes. Furthermore, an extension to the Bitcoin protocol reduces the required bandwidth of block propagation by replacing complete transactions in blocks by short transaction IDs [17]. This is possible because most transactions have been previously received by peers through the transaction propagation process. Transactions that have not been previously received can be requested subsequently from the remote peer.

Flooding vs. Gossip: When a message has been received or created, the client has to decide to which of its neighbors the message is relayed. Messages can be relayed either to all neighbors (flooding), or to a (randomly selected) subset of neighbors. This decision mainly affects the requirements performance, cost of participation, and DoS resistance, but can also affect anonymity.

Discussion: Flooding has a higher bandwidth cost compared to gossiping. However, when using an announce-and-request protocol, the bandwidth cost does not increase linearly in the number of selected neighbors. With an increasing number of selected neighbors the probability that these neighbors already

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

have received the message increases, thus eliminating the need to relay the message itself.

Not relaying messages to certain neighbors has the same effect as not having a connection to these neighbors. Therefore, it has the same negative effect on DoS resistance as a small number of connections in a very dynamic network (i.e., with quickly changing connections). As the subset of neighbors is randomly selected (e.g. for each message), there is a risk that certain messages do not propagate through the whole network. How large that probability is, given a certain network topology and relay probability has been analyzed for decades in the field of epidemic spreading (e.g., [66]).

Examples: The Bitcoin client uses flooding. The first phase of the proposed Dandelion strategy relays messages to one neighbor peer only, which can be seen as a gossip strategy [77]. It was also proposed to route messages along certain paths to miners [27].

Relay Delay: Clients can not only decide to which neighbors they relay messages, but also when. This decision mainly affects the requirements performance, anonymity, and topology hiding.

Discussion: The decision when to relay a message can be deterministic or probabilistic. An example for a deterministic approach would be a time-slotted system which rebroadcasts all messages received within a certain time slot at the end of that slot. An example for a probabilistic approach would be a system where upon reception of a message a random delay according to some probability distribution is used to calculate the sending time.

Deterministic decisions might be better from a performance perspective in cases where strict bounds on the information propagation delay are required. For these cases (e.g., real-time requirements), however, the considered systems are in general unsuitable. Indeterminism can be beneficial for topology hiding and anonymity as the attacker has incomplete knowledge of the probabilistic process and can only model the used probability distribution and not the outcome of each single decision. Obviously, deliberately delaying messages reduces the propagation speed, i.e., there is an inherent tradeoff between performance on the one side and topology hiding and anonymity on the other side. A more detailed analysis is presented in Section VII.

Examples: Bitcoin delays the relaying of messages according to an exponential distribution.

Message Accumulation: So far, we have only considered one message at a time and considered several messages and their relaying times and neighbors to be independent. However, a client could aggregate multiple messages so that they are sent at the same time, which can reduce bandwidth costs by reducing the required control data (e.g., packet header). It can also affect the requirements performance, anonymity, and topology hiding.

Discussion: Message accumulation only makes sense when clients do not rebroadcast messages immediately, because only then the collection of messages is possible. Besides reducing bandwidth costs, it can be more efficient from a software engineering perspective to maintain only one queue per neighbor with one potential sending time instead of maintaining one sending time per message.

Message accumulation can reduce the average relay delay depending on the number of aggregated messages: messages that enter a queue that already contains many messages may be sent early because their time of sending has already been scheduled before they have been received by the client. Message accumulation might improve topology hiding and anonymity because it adds more entropy that is unknown to the attacker as sending times depend on message receptions the attacker does not know about. However, the accumulation of messages might also enable new fingerprinting possibilities. Active adversaries can, for example, regularly send new messages to a remote peer and can infer the time the remote peer received a certain message from another peer based on the other messages that are within the same aggregated set. No extensive analysis of these fingerprinting possibilities has been published so far.

Examples: Bitcoin uses message accumulation by maintaining one queue of outgoing messages per neighbor. Messages to be relayed to that neighbor are appended to the queue and the sending time of all messages in the queue is determined when the queue was previously flushed.

C. Remarks

Based on the implementation of real-world permissionless blockchains we structured the design space of the network layer into several aspects. Fig. 3 sketches the effects of exemplary design choices on the fulfillment of requirements. Fig. 3 can be read in several ways: First, assuming a fixed set of design decisions (e.g., taken from an existing system), one can qualitatively assess the fulfillment of each requirement. Secondly, assuming a fixed importance of each requirement (e.g., based on the envisioned application during the design of the network layer), one can derive design decisions that support the important requirements. Furthermore, the figure shows which design options can be enabled by relaxing certain requirements. For instance, if anonymity is not required in a certain scenario, the relay delay might be shorter, which enhances performance and DoS resistance.

A more general observation can be made regarding the prevailing tradeoffs between requirements: With the exception of the aspect *topology generation*, all sketched design decisions either benefit the requirements performance and DoS resistance *or* the requirements topology hiding, cost of participation, and anonymity. This also implies that there are only few tradeoffs between performance and DoS resistance and between topology hiding, cost of participation, and anonymity. Roughly speaking, achieving performance and DoS resistance requires peers to send more data, whereas achieving anonymity, a low cost of participation, and topology hiding requires peers to send less data.

While only two design choices per aspect are sketched in Fig. 3, design choices are typically not binary. For instance, the number of connections can be anywhere between one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

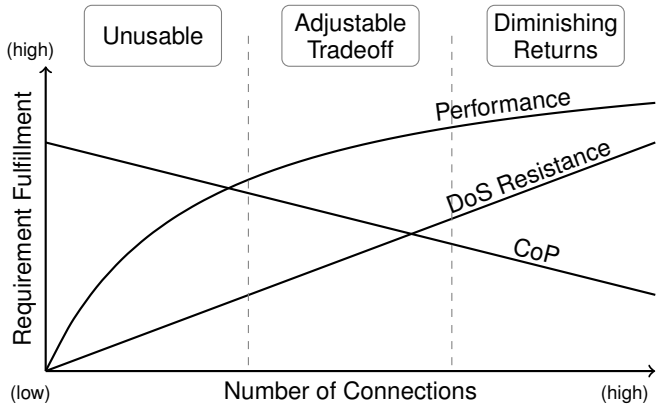

Fig. 4. Qualitative effect of the the *number of connections* on the requirements performance, DoS resistance and cost of participation (CoP).

and the total number of reachable peers. Fig. 4 depicts the effect of the number of connections on the fulfillment of the requirements performance, DoS resistance, and cost of participation. Because a minimum number of connections is required for the network to be connected (i.e., to have a path between any two peers), any design with less than that number becomes unusable (i.e, performance and DoS resistance are not sufficient). While increasing the number of connections substantially enhances performance for a small number of connections, the effect diminishes with a higher number of connections. On the other hand, the cost of participation increases linearly with each new connection. Therefore, there is a certain range in which the inherent tradeoff should be adjusted to satisfy the requirements.

While we discussed each aspect individually, there are interdependencies between certain aspects. Obvious dependencies are the available information sources and how the information is used (e.g., increasing the number of connections based on anomaly detection, increasing the relay delay based on network statistics). In order to comprehensively assess the fulfillment of a requirement in a certain design, all aspects that affect a requirement (cf. Table I and Fig. 3) need to be be considered.

Finally, the design of the network layer is, with few exceptions, only limited by the creativity of the designer. Hence, while covering a wide range of aspects, no claim of completeness can be made. Having described the design space and implications of design decisions facilitates the detailed, quantitative analysis of certain aspects as demonstrated in the following sections.

VI. In-Band Peer Discovery

In the previous section we discussed a large number of aspects and qualitatively discussed tradeoffs associated with each aspect. For most aspects, no models exist in the literature that allow a quantitative analysis of the effect of certain design decision on the requirements. Motivated by this lack of models, we pick two aspects to demonstrate directions of future research. In this section, we explore how the design of the in-band peer discovery strategy affects the requirements topology hiding, DoS resistance, and performance of a network, and demonstrate a method for quantitatively assessing the quality of a peer discovery strategy.

A. Strategy Requirements & Tradeoffs

It is important that a client is able to quickly establish outgoing connections, not only from a performance perspective but also for DoS resistance. In the event of an eclipse attack a client should be able to react on the loss of connections caused by the attack by the establishment of new connections. Metrics that reflect this ability are the total number of IP addresses and the number of reachable addresses in a client's address list. The total number of reachable addresses gives an upper bound on the number of successful connections a client can establish, the share of reachable addresses indicates the probability of successfully establishing a connection per connection attempt, assuming the client tries to connect to randomly chosen addresses.

Another requirement that is affected by the peer discovery strategy is topology hiding as an adversary can infer connections between peers based on the address messages peers send to their neighbors [56]. Metrics that indicate the success of an adversary are precision and recall for the classification problem of whether a direct connection between two peers exists.

Intuitively we expect that the choice of parameters of the peer discovery strategy has an oppositional effect on the two requirements. For instance, a configuration that sends a large number of IP addresses at short time intervals with precise timestamps of connected IP addresses will result in a good DoS resistance, but will also make it easy for adversaries to infer the network topology. We will quantitatively analyze the tradeoff between these two requirements in the next sections.

Finally, adversaries should be unable to eclipse peers by filling their address list with IP addresses under the adversary's control and making the victim peer connect exclusively to attacker's peers [41]. We will discuss this requirement in Section VI-F.

B. Strategy Description

We analyze a basic in-band peer discovery strategy that periodically exchanges reachable IP addresses between connected peers.

Every client maintains an *address list* l containing tuples consisting of an IP address a_i and an associated timestamp t_i ($l = \{(a_1, t_1), (a_2, t_2), ...\}$). Every δ_s seconds a client sends an *address message* containing n randomly (uniform) selected entries of its address list to each neighbor. On reception of such a message from a neighbor a client updates its own address list: new addresses (and their timestamp) are added to the list, and the timestamp of known addresses is updated if a newer timestamp than the one stored is received. When a new connection is established to or from a client, the client adds the foreign IP address to its address list l and randomly selects

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

a timestamp for that IP address from a uniform distribution $\mathcal{U}[t-\delta_d, t]$ where t is the current time and δ_d a configured value. When the timestamp of a connected peer becomes smaller than $t-\delta_d$, a new timestamp is set according to the same uniform distribution ($\mathcal{U}[t-\delta_d, t]$). This strategy ensures that the timestamps of all connected peers are always newer than the current time minus δ_d, hence δ_d represents the maximum age of connected IP addresses in a peer's address list. Finally, addresses with timestamps smaller than $t-\delta_x$ are removed from a client's address list, with δ_x being a configurable parameter.

The described strategy is very simple and can be configured using only the parameters $n, \delta_s, \delta_d, \delta_x$. However, there are many more changes possible, e.g., the timestamp of connections could follow other probability distributions than the used uniform distribution, the subset of addresses to be sent to neighbors could be biased based on the timestamp, or the number of addresses to be sent could depend on the total number of entries in a client's address list or the connection duration. However, we will limit our analysis to the described strategy with its parameters and leave a more detailed assessment as future work.

C. Adversary Model

The adversary wants to infer the topology of the network based on information leaked by the peer discovery mechanism. For now we will assume a passive monitor adversary that establishes connections to peers and receives the announced addresses from its neighbors. A discussion of other adversary models is made in Section VI-F. We assume that the adversary knows all chosen parameters as well as all required parameters of the network (e.g., node degree distribution). Consider the example of an adversary that wants to know whether two peers, p_1 and p_2, are directly connected. Let us consider the case that the adversary is connected to p_1 only. One observation $o \in O$ by the adversary is the reception of one address message from p_1 containing a subset of p_1's address list. One observation can be either the age of the IP address of p_2 or the fact that the IP address of p_2 is not contained in the sent list i.e., $o \in \mathbf{R}^+ \cup \{\perp\}$ ($o = \perp$ implying that the IP address of p_2 is not contained in the sent list).

Let C be the random variable modeling the existence of a connection between two peers (i.e., $C = 1$ if both peers are directly connected, $C = 0$ otherwise). The maximum likelihood estimator (MLE) for a set of observations O maximizes the likelihood function

$$L(C = c|O) = P(C = c) \cdot \prod_{o \in O} P(o|C = c)$$

for $c \in \{0, 1\}$. In order to utilize the MLE, an adversary requires knowledge of the probability distributions $P(C = c)$ (i.e., the a-priori probabilities of two peers being connected) and $P(o|C = c)$ (i.e., the probability of making a specific observation o conditional to both peers being connected or not connected, respectively). Combining knowledge about the client source code with statistic properties of the network into a simulation model allows approximation of both probability distributions for real-world systems.[18]

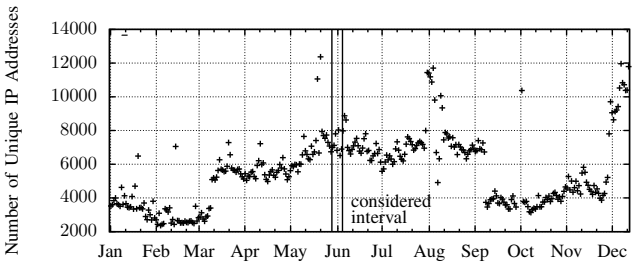

Fig. 5. Measured number of unique IP addresses of the Bitcoin P2P network to which connections were established per day during the year 2017.

D. Methodology

In order to analyze the discussed peer discovery strategy, we implemented a model of a P2P network as a discrete event simulation.[19] The simulation model has three types of events: a peer joins the network, a peer leaves the network, and a peer sends an address message to a neighbor. The churn of the network (i.e., join and leave events) was taken from a real-world measurement on the Bitcoin network: our monitor peer establishes connections to all reachable peers on the Bitcoin network. Every new connection to the monitor peer translates to a *join* event, every disconnect translates to a *leave* event. The simulation was performed with a one week snapshot from May 29th, 2017 until June 5th, 2017. During that period, our monitor peer established connections to 35,000 unique IP addresses. On average around 9,000 peers were concurrently reachable during that period, which is also the size of the simulated network. All simulations were performed for the duration of one simulated week.

Although the use of a specific snapshot from the Bitcoin network for parametrization of the simulation reduces generality of the obtained results, the selected snapshot is a representative sample of the Bitcoin network in 2017: Fig. 5 shows the number of unique IP addresses to which connections were established per day during the year 2017. While there were anomalous events during that year (e.g., several thousand Sybil peers on August 1st), no such event occurred during the considered time frame. Furthermore, the observed connection duration distribution during that time frame is consistent with the distributions observed during most of the year. Finally, other measurements of the Bitcoin network[20] are in corre-

[18] $P(o = \perp|C = 1)$ is calculated by dividing the parameter n (the number of IP addresses a client sends) by the total number of addresses in the client's address list. Both values can be approximated by the adversary from the client source code. $P(o = \perp|C = 0)$ is calculated as $P(o = \perp|C = 1)$ multiplied by the probability that a client has the IP address in question in its list. An adversary with knowledge of the client source code and basic statistic properties of the network (e.g., number of nodes, churn) can approximate that probability by simulation. That way, an adversary can also derive $P(o|C = 1)$ and $P(o|C = 0)$ for $o \neq \perp$.

[19] https://github.com/tillneu/peerdiscovery-sim

[20] Publicly available data sources include https://bitnodes.earn.com/ and http://bitcoinstats.com/network/propagation/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

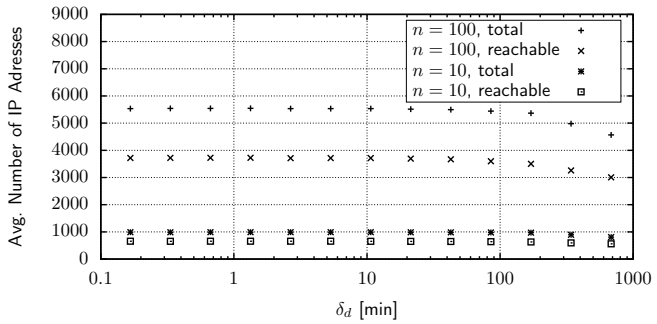

Fig. 6. Average number of IP addresses in all client's address lists.

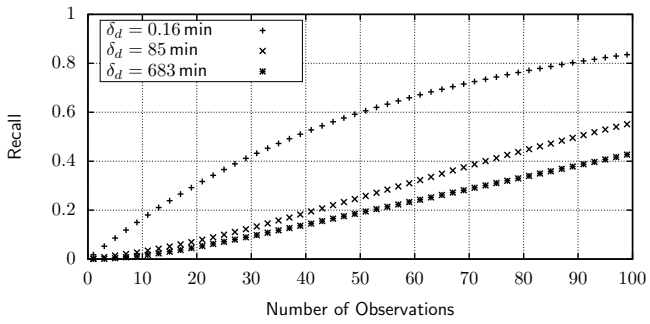

Fig. 7. Average recall depending on the number of observations $|O|$.

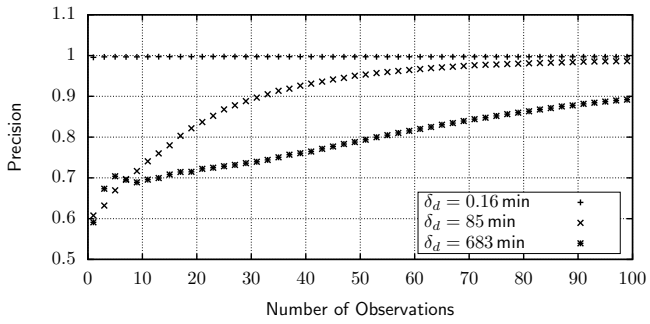

Fig. 8. Average precision depending on the number of observations $|O|$.

spondence to our measurements. We will discuss whether the results can be generalized to other networks in Section VI-F4.

When a peer joins the network in the simulation, it establishes 8 outgoing connections to randomly selected peers. When a peer leaves the network, other peers that established an outgoing connection to the leaving peer establish new outgoing connections to other randomly selected peers so that the total number of outgoing connections remains 8. This behavior matches the behavior of the Bitcoin client.

The *address send* event implements the described strategy. The simulation does not account for latencies between peers because these are typically in the milliseconds range and, therefore, irrelevant for our analysis. We also ignored unreachable peers during the simulation but will discuss the effect of unreachable peers on the results later. The simulation also allows us to directly observe all probability distributions (cf. Sec. VI-C) required for the adversary. We simulate the adversary by providing all these probability distributions and observations to the simulated adversary and letting the adversary guess whether peers are connected or not. This implies that our adversary has perfect knowledge, which is almost impossible to achieve in reality.

E. Results

We will now first discuss simulation results regarding DoS resistance and then discuss results regarding topology hiding. We fixed the address send interval δ_s to one hour and the address deletion age δ_x to 24 hours for all simulation runs. Parameters that were varied were the number of addresses to send n and the maximum age of connected IP addresses δ_d.

Fig. 6 shows the average number of IP addresses (total and reachable only) in all client's address lists for $n \in \{10, 100\}$ depending on δ_d. For $n = 100$ the average total number of IP addresses is around 5,500 for $\delta_d < 100$ minutes. The average number of reachable IP addresses is around 3,700 for $\delta_d < 100$ minutes. For very large choices of δ_d, the total and reachable number of IP addresses decline. With $n = 10$ a similar behavior can be seen, although the overall numbers are much smaller (less then 1,000).

The results indicate that the discussed peer discovery strategy works well over a wide range of parameter choices. Even when configured to sending only 10 addresses per hour per neighbor ($n = 10$), the address list of peers still contains around 600 reachable IP addresses on average. Furthermore, the effect of the choice of δ_d is negligible for $\delta_d < 100$ minutes. Please note that the given values are averages over all peers after one week of simulated time. Peers that remain in the network for a long time and have established many connections have more entries in their address list than peers that joined the network just a short time ago. Furthermore, the session length distribution of peers affects the share of reachable IP addresses: a large number of short living peers would increase the total number of IP addresses but would contribute less to the number of reachable IP addresses.

Based on these results we chose three parameter sets for analysis of a topology inference attack. We fixed the number of addresses to send to $n = 100$ and simulated a topology inference attack for $\delta_d \in \{0.16\,\text{min}, 85\,\text{min}, 683\,\text{min}\}$. $\delta_d = 0.16\,\text{min}$ and $\delta_d = 683\,\text{min}$ represent extreme choices (corresponding to the outermost points in Fig. 6), $\delta_d = 85\,\text{min}$ corresponds to a choice where, based on the results from Fig. 6, no significant deterioration of DoS resistance can be expected.

Fig. 7 shows the expected recall depending on the number of observations $|O|$. Recall is defined as the quotient of true positives and relevant elements, i.e., the share of existing connections the adversary infers. As expected, the recall rises in the number of observations and also rises with decreasing δ_d.

Fig. 8 shows the expected precision depending on the num-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

ber of observations $|O|$. Precision is defined as the quotient of true positives and all inferred elements, i.e., the share of inferred connections that actually exist. For $\delta_d = 0.16\,\mathrm{min}$ the precision is close to 1 regardless of the number of observations. For the other δ_d, the precision increases with the number of observations.

F. Lessons Learned, Discussion & Open Issues

We will first discuss the implications of the shown results before we address various aspects of the adversary model and the discussed strategy. Finally, we discuss the used methodology in general.

1) Results: Assume a configuration with $\delta_d = 85\,\mathrm{min}$ and $n = 100$. Our results then show that an adversary that is able to obtain 100 observations from a certain peer is able to identify 55 % of the neighbors of that peer with very high precision (i.e., almost no false positives). With the setting of δ_s to one hour, an adversary that only connects to that peer would have to wait 100 hours. A monitor adversary that connects to all reachable peers would only need to wait 50 hours to obtain 100 observations for each pair of peers because he receives one address message per hour from every peer. By using more than one monitor peer an adversary can easily reduce the required time to collect the desired number of address messages.

One important requirement for the adversary is that the network topology does not change while making observations. Measurements on the Bitcoin network show that roughly 40 % of all connections exist for periods longer than one week.[21] In these cases even an adversary with one monitor peer would be able to collect enough address messages to identify a substantial share of connections of peers.[22]

2) Adversary Model: The adversary model we considered in this analysis is in some aspects stronger than what we expect a real adversary to be (e.g., perfect knowledge of all relevant parameters and distributions), however, there are other aspects where a stronger adversary model might be appropriate. First, we assumed the adversary to be passive. In reality, an adversary can easily actively transmit its own address messages to other peers.

A very simple attack is to flood unreachable IP addresses to other peers so that the share of reachable IP addresses decreases. The number of IP addresses an adversary can send per connection is limited by the choice of n/δ_s and the expiration duration δ_x. For instance, with δ_s and δ_x as before and $n = 100$, an adversary can send up to 2400 unreachable IP addresses per connection in 24 hours. Even with 100 adversarial connections, the share of reachable IP addresses would still be around 1.5 %, implying that a connection can be established within a few minutes or less. Furthermore, the required memory to store all addresses would be less than 10 MB. Hence, this attack is not viable for most scenarios.

A known attack is to eclipse peers by filling their address list with IP addresses under the adversary's control and making the victim peer connect exclusively to attacker's peers [41]. If a client selects IP addresses from its address list randomly for establishing connections, the adversary has no other option than to announce the IP addresses of his peers and hope that the victim peer connects exclusively to his peers. In the analyzed case, an adversary has to spawn more than 40,000 peers and announce their IP addresses in order to eclipse a peer with 8 outgoing connections with a probability of 50 %.[23]

Furthermore, the considered MLE is only optimal assuming independence between several observations. Adversaries could exploit further information such as the differences in timestamps of certain IP addresses between several observations. Also, partial knowledge of the network topology might enable an adversary to not only exploit information received from peers p_1 and p_2 in order to infer whether a connection between p_1 and p_2 exists, but also to facilitate information sent by other peers (e.g., the neighbors of p_1 and p_2).

3) Strategy: Although the analyzed peer discovery strategy is very simple, the results indicate that an adequate parametrization of the strategy could already satisfy the considered requirements to a reasonable extend. In scenarios with stronger requirements, several changes to the strategy are possible: In order to prevent too many connections to peers from small IP ranges or the same AS, simple checks can be implemented after the random selection of IP addresses from the address list. Selecting IP addresses biased towards new addresses might improve performance, however, it can be easily exploited by adversaries to increase chances of connections to its own peers. Therefore, a random selection is preferable. A possibility to improve performance and DoS resistance at the cost of higher bandwidth cost proposed in [41] and implemented in Bitcoin[24] is to continuously check whether IP addresses are reachable. This drastically reduces the share of unreachable IP addresses.

The fact that the topology of a changing network is harder to infer suggests the idea to deliberately disconnect from neighbors after a certain period and regularly establish new connections to other peers.[25] If topology hiding is considered extremely important in a certain system, clients could apply a notion similar to the *privacy budget* known from differential privacy [25] to its own connections. Sending messages that allow an adversary to infer that connection decreases the budget of the connection. When the budget is exceeded, the connection is terminated. One drawback of this approach is, however, that an instable network topology not only increases bandwidth cost, but also reduces DoS resistance against short term Sybil attacks. An adversary that enters the network with a large number of Sybil peers is able to *thin out* connections between honest peers much faster in a changing network,

[21] https://dsn.tm.kit.edu/bitcoin

[22] Although some parameters were chosen in accordance with those in Bitcoin, our results are not directly applicable to the Bitcoin network because neither the discussed peer discovery strategy nor the network topology matches the ones in Bitcoin.

[23] Calculated as: (number of adversarial peers / total number of reachable IP addresses in l)$^8 = 0.5$: $40000/43700^8 \approx 0.49$.

[24] https://github.com/bitcoin/bitcoin/blob/5114f8113627791b871c88998bd/src/net.cpp/#L1756

[25] This idea has been proposed for structured P2P networks [15].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

as peers will establish connections to the Sybil peers faster. Therefore, a thorough analysis of all implications of such an approach is required.

4) Methodology & Lessons Learned: The used simulation based methodology enables a precise analysis of highly irregular systems with possibly complex strategies. However, the results we obtained are valid only for the considered model, which consists of a network parametrization obtained from snapshots of the Bitcoin P2P network, and a basic in-band peer discovery strategy. If the results should be applicable to a real-world system, the simulation model has to be parametrized so that it matches the real system. For instance, if the real-world system has a much higher churn with peers being reachable for much shorter durations than peers on the Bitcoin network, this parameter has to be modeled accordingly. Obviously, the results for such a network can differ vastly from the results presented here.

One typical challenge in the process of modeling real-world systems is the lack of precise estimations for certain parameters. Even though some parameters are known (e.g., behavior from the client source code) and others can be easily measured (e.g., number of reachable peers), other parameters are hard to measure and need to be estimated (e.g., the number of unreachable peers). Actually, the networks created by permissionless blockchains are designed to hide certain parameters in order to enhance anonymity and topology hiding. We emphasize that the lack of a precise estimation for a parameter does generally not prevent meaningful simulation results. First, the effect of an unknown parameter on the results might be small. Simulations enable straightforward sensitivity analysis by variation of the parameter in question [43]. Furthermore, often reasonable assumptions can be made that make a precise estimation of a parameter not required. For instance, in our analysis we chose to set the number of unreachable peers to zero for two reasons. First, a client could check whether a neighbor is actually reachable before adding the neighbor's IP address to the local address list. This would completely eliminate any effect of unreachable peers on any of the analyzed metrics. Secondly, even if clients do not perform such a check, only the average total number of IP addresses in client's address lists would increase (cf. Fig. 6). This would reduce the share of reachable IP addresses linearly in the number of unreachable peers.

Another observation, which can be made from the analysis of the peer discovery mechanism, is that often more than one modeling approach is required in order to model the system. For instance, the main model we use is a discrete event simulation model which captures the behavior of peers. However, we also have to model the adversary (within the simulation), which is done using a maximum likelihood estimator, which requires a closed form stochastic model. During such a research process, errors can be made using any chosen method (e.g., implementation errors in the simulation, making false independence assumptions in stochastic models). One advantage of applying multiple methods is the possibility to cross validate each model (and implementation) with the other models. While this can be challenging, if the methods are at vastly different abstraction levels, one can reduce this abstraction gap by using additional methods. For instance, simple Monte Carlo simulations turned out to be very effective at detecting errors in realistic discrete-event simulations, as well as in analytical models.

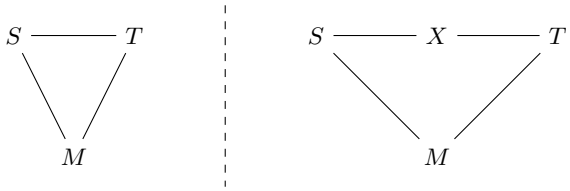

Fig. 9. Considered scenario: The adversary M wants to infer whether S and T are directly connected (left side), or whether S and T are not directly connected (right side).

VII. Communication Strategy: Relay Delay

We already emphasized the lack of quantitative models for the analysis of most of the aspects discussed in Section V. While quantitative models can be used for the analysis of existing strategies, such models can also be utilized to find optimal parametrizations for certain strategies. In this section we give an example that shows how the relay delay of the communication strategy can be optimized for topology hiding and performance. Our approach optimizes the relay delay in a simplified scenario, and serves as an example for the underlying optimization problem.

A. Strategy Requirements & Tradeoffs

It is important that messages are disseminated quickly through the network, in order to reach consensus on the transmitted information. Therefore, any deliberate delay of messages negatively affects performance. On the other hand, immediately rebroadcasting messages enables attacks on the anonymity of users and also allows inference of the network topology. Therefore, the chosen relay delay should balance the system's requirements regarding performance on the one hand, and anonymity and topology hiding on the other hand.

B. Strategy Description

We consider a simple relay delay strategy that delays every message independently using a given delay function d. Messages are not accumulated, messages are directly pushed, and no changes to the delay function d are made during runtime. All notation in this section is discrete, hence d is a probability mass function (PMF) that defines the probability that a message is delayed by a certain duration. The duration is a discretized representation of time, e.g., time slots of millisecond precision. In this section we are interested in finding a delay function d that optimizes topology hiding while still providing a certain performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

C. Adversary Model

We consider the simplified scenario sketched in Fig. 9: An adversary M is connected to two peers S and T and wants to infer, whether S and T are directly connected or not. The adversary creates a message (e.g., a transaction) and sends it to S so that S receives the message at time 0. The adversary then waits and measures the duration δ until T sends the message to M. δ is therefore the delay from S receiving the message until T sends the message to M.

Consider the case that the latency between any two peers is one time unit, i.e., the transmission of a message over one link takes one time unit, and each peer immediately rebroadcasts each message to its neighbors. Then, the adversary knows that if $\delta = 2$ then S and T are directly connected, if $\delta = 3$ then S and T are not directly connected. In reality, the latency between two peers is not constant but follows a probability distribution. We assume that the latency between any two peers follows the same distribution $\lambda(t)$, which is known to the adversary.

We will now formulate a maximum likelihood estimator that can be used by the adversary to infer whether a direct connection between two peers exists. Let $f * g$ denote the convolution of the (discrete) functions f and g, and let f^{*n} denote the n-th convolution power of a function f. Let $H \in \{1, 2\}$ be the random variable modeling the hop-distance between S and T. $H = 1$ if S and T are directly connected, $H = 2$ if there is one hop between S and T. The resulting distribution for the overall delay δ equals time t conditional to the hop distance H is then given by

$$P(\delta = t|H = h) = (\lambda^{*h} * d^{*(h+1)})(t). \quad (1)$$

For example, if S and T are directly connected ($H = 1$), the message is delayed by one link latency (the link between S and T) and two relay delays according to d (at peers S and T).

The probability that the distance between S and T is h, given an observed time difference of t, is given by

$$P(H = h|\delta = t) = \frac{P(\delta = t|H = h) \cdot P(H = h)}{P(\delta = t)}. \quad (2)$$

$P(\delta = t|H = h)$ can be calculated using equation (1), $P(\delta = t)$ can be calculated using the law of total probability, $P(H = h)$ is assumed to be known to the adversary based on statistic properties of the network. The MLE maximizes $P(H = h|\delta = t)$, i.e., the adversary guesses the hop-distance H that is most likely based on the observation δ.

D. Methodology

Based on the adversary model we will now derive a delay function d that maximizes the expected error of the adversary, i.e., which makes topology inference as hard as possible. The expected error e_d of the guess of the adversary depends on the delay function d, and can be calculated using equation (2) as

$$e_d = \sum_t [P(\delta = t) \cdot (1 - \max_h P(H = h|\delta = t))]. \quad (3)$$

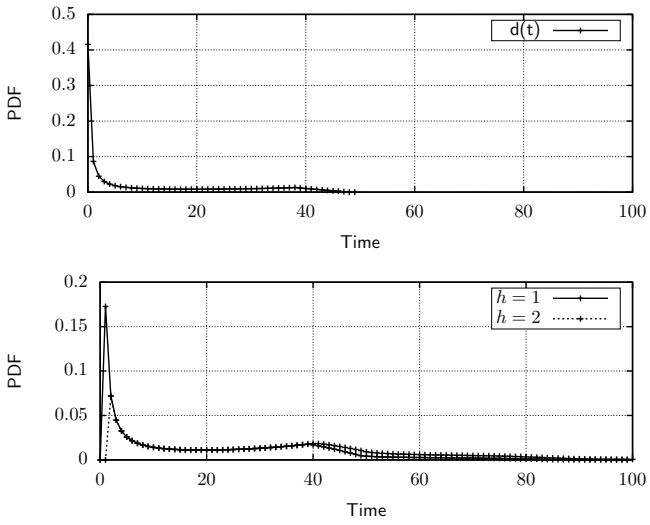

Fig. 10. Top: Optimal $\hat{d}$. Bottom: Resulting $P(\delta = t|H = h)$ for $h \in \{1, 2\}$. Parameters: $\mu = 10$, $\lambda(1) = 1$ (0 else), $P(H = 1) = P(H = 2) = 0.5$

The expected error is the objective function that should be maximized. The variable in the optimization problem is not a scalar, but the delay function d. The optimization has to ensure that d is a PMF (i.e., $d(t) \geq 0 \forall t, \sum_t d(t) = 1$). Furthermore, we want to limit the expected value $\mathbb{E}(d)$ of d to be less than some constant μ. The choice of the parameter μ reflects performance constraints in the system: A small choice of μ ensures fast message propagation, a large μ allows for slower message propagation. The resulting optimization problem is

$$\begin{aligned} \underset{d}{\text{maximize}} \quad & e_d \\ \text{subject to} \quad & \mathbb{E}(d) < \mu \\ & d \text{ is a PMF.} \end{aligned} \quad (4)$$

Because d is a discrete function, (4) is a multidimensional optimization problem, where each time step of the delay function d is one dimension (i.e., one free variable) of the optimization problem. The optimization problem is also constrained ($\mathbb{E}(d) < \mu$ and d is a PMF). However, the optimization problem with both constraints can be transformed into an unconstrained optimization problem (e.g., using gradient projection [69]). The optimal solution to the transformed problem can be approximated using common software for optimization (e.g., we used the BFGS search algorithm implemented in the Dlib toolkit [49]).

E. Results

We assume a scenario with a fixed latency of one time unit between all directly connected peers ($\lambda(1) = 1, \forall t \neq 1 : \lambda(t) = 0$), equal a priori probabilities ($P(H = 1) = P(H = 2) = 0.5$), and a maximum expected value of 10. The top part of Fig. 10 shows the approximated optimal delay function $\hat{d}(t)$. For a delay of 0 the probability peaks at around 0.42 and rapidly declines to less than 0.02, where it stays until t=39. The expected value of $\hat{d}$ is 10, $e_{\hat{d}}$ is 0.41. Please note that $\hat{d}$ does

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

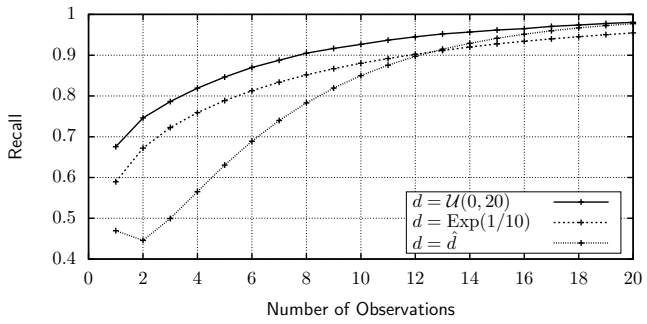

Fig. 11. Recall depending on the number of observations for $d \in \{\hat{d}, \text{Exp}(1/10), \mathcal{U}(0, 20)\}$.

not decrease monotonically over time and $\hat{d}$ does not resemble any common PMF (e.g., the PMF of a binomial distribution).

The bottom part of Fig. 10 shows $P(\delta = t|H = h)$ for the same scenario with the optimal function $\hat{d}$ for $h \in \{1, 2\}$. $P(\delta = t|H = h)$ is used by the MLE to derive the probability for $H = 1$ and $H = 2$, based on the observed time difference δ. We can see that $P(\delta = t|H = 1)$ and $P(\delta = t|H = 2)$ are exactly congruent between $\delta = 2$ and $\delta = 19$. This implies that an observation within that range is completely useless for the adversary. However, if the adversary observes $\delta = 1$, he can be sure that $H = 1$ because $P(\delta = 1|H = 2) = 0$ and $P(\delta = 1|H = 1) > 0$. If the adversary observes $\delta \geq 40$, he learns that $H = 2$ is more likely than $H = 1$ because $P(\delta = t|H = 2) > P(\delta = t|H = 1)$.

Especially the observation of $\delta = 1$ is valuable for topology inference, because it allows the definite conclusion that both remote peers are directly connected. While $P(\delta = 1|H = 1)$ is only around 17 % for one single observation, the probability that at least one out of ten observations for two directly connected peers is $\delta = 1$, is already at 85 %. This property makes $\hat{d}$ a non-optimal delay function when the adversary combines multiple observations.

Fig. 11 shows the recall (i.e., the probability that an existing connection is correctly inferred) depending on the number of observations for d being $\hat{d}$, a uniform distribution, and an exponential distribution (all with a mean of 10), obtained by simulation. Although $\hat{d}$ results in the lowest recall for a small number of observations, for large numbers of observations other distributions are better for topology hiding.

All shown delay functions result in a recall of more than 95 % for 20 observations. However, these values are a result of the strong adversary model with perfect knowledge of all network properties (e.g., latency, node degree distribution). Imperfect estimation of these properties causes a decline in inference quality [63].

F. Lessons Learned, Discussion & Open Issues

Although this example is very limited, it shows the potential of a formalization of a design decision's tradeoffs as an optimization problem and also indicates future research possibilities. Finding optimal solutions for larger number of observations requires a reformulation of the optimization problem so that the objective function accounts for the larger number of observations (e.g., minimize the expected recall after n observations). Because a numerical calculation of the expected recall is intractable even for small numbers of observations, the optimization problem might be formulated continuously and solved analytically, e.g. using calculus of variations [33].

Furthermore, the considered scenario does not account for realistic latency distributions and network topologies. Finally, the optimization problem only covers the tradeoff between performance and topology hiding, but does not incorporate the requirement anonymity. With clearly formulated requirements, a multicriterial optimization problem could result in design decisions that are optimal with regard to all affected requirements.

We will now discuss possible alternatives to the used research method and insights gained from the execution of the approach. Many methods could be used to optimize the relay delay mechanism, ranging from machine learning to completely analytical approaches. For instance, we considered the use of simulation based optimization [39], i.e., use simulations to calculate the objective function. This would enable us to consider scenarios with more complex network topologies and multiple messages. However, we had to discard the idea, because in the considered optimization problem, changes to the parameters have a very small effect on the objective function. This is not a problem when the objective function can be calculated accurately, however, it becomes a problem when the objective function is approximated by probabilistic simulations. In that case, the error induced by the probabilistic nature of the simulations becomes greater than the changes caused by the change of the parameters by the optimization algorithm. This impedes the optimization, because the algorithm relies on accurate measures of the effect of parameter changes on the objective function. In order to improve the accuracy, the simulation run time would have to be increased to unbearable time spans for the considered scenario.

With the use of an analytical method comes the need for more abstraction and tighter system boundaries. Fig. 12 visualizes this relationship between the degree of abstraction of research methods and the need for parameter and behavior assumptions. For instance, in Section VI a discrete event simulation is used, which requires a parametrization based on measurements of the real-world Bitcoin network. Furthermore, the adversary in Section VI is modeled analytically, and used *within* the simulation. Contrary, the behavior model used in this section is purely analytical, and the parametrization is based on simplifying assumptions (e.g., latencies are assumed to be constant).

VIII. Conclusion & Directions of Research

The systematization of a large number of attacks showed that many aspects of the network layer of deployed blockchains were vulnerable in the past. Based on these threats we identified the security requirements anonymity, DoS resistance,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2018.2852480, IEEE Communications Surveys & Tutorials

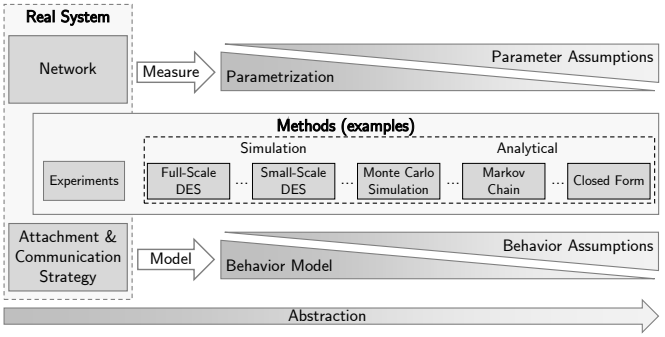

Fig. 12. Research methodology for the analysis of the network layer of permissionless blockchains.

and topology hiding, and the requirements performance and low cost of participation. Furthermore, we surveyed the design space of the network layer and qualitatively analyzed the effect of design decisions on requirements. On the one hand, this survey enables designers of blockchains to come to more justified design decisions, which balance the requirements of the application. On the other hand, this survey reveals future research directions. The analysis of the in-band peer discovery and the relay delay mechanisms indicated that the presence of inherent tradeoffs does not prevent designs that sufficiently balance the affected requirements.

Most of the discussed **network layer aspects** are far from being well understood. For some aspects, the design options seem clear, but the effect of each option cannot be given quantitatively, e.g., the effects of message accumulation on topology hiding and anonymity are not clear. For other aspects, the design space should be further explored, e.g., exploiting additional information sources (e.g., sidechannel information from other clients, connection anomaly detection) might improve DoS resistance and performance.

Despite the large number of known attacks on the network layer, only few **quantitative models** exist that describe the effect of design decisions and adversary capabilities on the fulfillment of requirements. In order to parametrize such models, more measurements of real-world systems and estimations of parameters are required. Furthermore, the (semiautomatic) generation of behavior models based on client implementations (e.g., [55]) could simplify and accelerate performing simulation based research. Finally, reusable and composable network models could enable the joint analysis of multiple aspects of the network layer.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) within the project *KASTEL_IoE* in the Competence Center for Applied Security Technology (*KASTEL*) and by the state of Baden-Württemberg through bwHPC. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[3] K.-J. Alm. Rate limiting via peer specified challenges (bip 154). https://github.com/bitcoin/bips/blob/master/bip-0154.mediawiki.

[4] M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 375–392. IEEE, 2017.

[5] R. Atat, L. Liu, H. Chen, J. Wu, H. Li, and Y. Yi. Enabling cyber-physical communication in 5g cellular networks: challenges, spatial spectrum sensing, and cyber-security. *IET Cyber-Physical Systems: Theory & Applications*, 2(1):49–54, 2017.

[6] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.

[7] A. Back. Hashcash-a denial of service counter-measure. 2002.

[8] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding*, pages 245–257. Springer, 2001.

[9] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with Bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–5. IEEE, 2013.

[10] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in Bitcoin P2P network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014.

[11] A. Biryukov and I. Pustogarov. Bitcoin over Tor isn't a good idea. *arXiv preprint arXiv:1410.6079*, 2014.

[12] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121. IEEE, 2015.

[13] F. Caccioli, G. Livan, and T. Aste. Scalability and egalitarianism in peer-to-peer networks. In *Banking Beyond Banks and Money*, pages 197–212. Springer, 2016.

[14] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[15] T. Condie, V. Kacholia, S. Sank, J. M. Hellerstein, and P. Maniatis. Induced churn as shelter from routing-table poisoning. In *NDSS*, 2006.

[16] M. Conti, C. Lal, S. Ruj, et al. A survey on security and privacy issues of Bitcoin. *arXiv preprint arXiv:1706.00916*, 2017.

[17] M. Corallo. Compact block relay (bip 152). https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki, 2016.

[18] A. de Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, May 2018.

[19] C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[20] S. Delgado-Segura, C. Pérez-Solà, J. Herrera-Joancomartí, G. Navarro-Arribas, and J. Borrell. Cryptocurrency networks: A new p2p paradigm. *Mobile Information Systems*, 2018, 2018.

[21] J. Dinger and O. Waldhorst. Decentralized bootstrapping of P2P systems: A practical view. *NETWORKING 2009*, pages 703–715, 2009.

[22] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[23] J. A. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí. The Bitcoin P2P network. In *International Conference on Financial Cryptography and Data Security*, pages 87–102. Springer, 2014.

[24] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[25] C. Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[26] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[27] O. Ersoy, Z. Ren, Z. Erkin, and R. L. Lagendijk. Information propagation on permissionless blockchains. *arXiv preprint arXiv:1712.07564*, 2017.

[28] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[29] M. Fadhil, G. Owenson, and M. Adda. Locality based approach to improve propagation delay on the Bitcoin peer-to-peer network.

[30] G. Fanti, P. Kairouz, S. Oh, K. Ramchandran, and P. Viswanath. Hiding the rumor source. *IEEE Transactions on Information Theory*, 2017.

[31] G. Fanti and P. Viswanath. Anonymity properties of the Bitcoin P2P network. *arXiv preprint arXiv:1703.08761*, 2017.

[32] H. Finney. The finney attack. https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384.

[33] I. M. Gelfand, R. A. Silverman, et al. *Calculus of variations*. Courier Corporation, 2000.

[34] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber. On the privacy provisions of bloom filters in lightweight Bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 326–335. ACM, 2014.

[35] A. Gervais, G. Karame, S. Capkun, and V. Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3):54–60, 2014.

[36] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[37] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun. Tampering with the delivery of blocks and transactions in Bitcoin. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 692–705. ACM, 2015.

[38] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[39] A. Gosavi. Simulation-based optimization. *parametric optimization techniques and reinforcement learning*, 2003.

[40] M. Hearn and M. Corallo. Connection bloom filtering (bip 37). https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki.

[41] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, 2015.

[42] M. Imani, A. Barton, and M. Wright. Forming guard sets using AS relationships. *CoRR*, abs/1706.05592, 2017.

[43] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.

[44] X. Jin and S.-H. G. Chan. Unstructured peer-to-peer network architectures. *Handbook of Peer-to-Peer Networking*, pages 117–142, 2010.

[45] K. Kanemura, K. Toyoda, and T. Ohtsuki. Design of privacy-preserving mobile bitcoin client based on γ-deniability enabled bloom filter. In *Personal, Indoor and Mobile Radio Communications(PIMRC), 2017. IEEE 18th International Symposium on*. IEEE, 2017.

[46] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in Bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917. ACM, 2012.

[47] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[48] M. C. K. Khalilov and A. Levi. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Communications Surveys & Tutorials*, 2018.

[49] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10(Jul):1755–1758, 2009.

[50] P. Koshy, D. Koshy, and P. McDaniel. An analysis of anonymity in Bitcoin using P2P network traffic. In *Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 469–485. Springer Berlin Heidelberg, 2014.

[51] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, 2013.

[52] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[53] Y. Marcus, E. Heilman, and S. Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. http://www.cs.bu.edu/~goldbe/projects/eclipseEth.pdf, 2018.

[54] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[55] A. Miller and R. Jansen. Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications. *IACR Cryptology ePrint Archive*, 2015:469, 2015.

[56] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering Bitcoin's public topology and influential nodes. https://cs.umd.edu/projects/coinscope/coinscope.pdf, 2015.

[57] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

[58] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[59] C. Natoli and V. Gramoli. The balance attack against proof-of-work blockchains: The r3 testbed as an example. *arXiv preprint arXiv:1612.09426*, 2016.

[60] K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.

[61] T. Neudecker. Bitcoin cash (bch) sybil nodes on the Bitcoin peer-to-peer network. http://dsn.tm.kit.edu/publications/files/332/bch_sybil.pdf.

[62] T. Neudecker, P. Andelfinger, and H. Hartenstein. A simulation model for analysis of attacks on the Bitcoin peer-to-peer network. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1327–1332, May 2015.

[63] T. Neudecker, P. Andelfinger, and H. Hartenstein. Timing analysis for inferring the topology of the Bitcoin peer-to-peer network. In *2016 Intl IEEE Conference on Advanced and Trusted Computing (ATC)*, pages 358–367, July 2016.

[64] T. Neudecker and H. Hartenstein. Could network information facilitate address clustering in Bitcoin? In *4th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 2017*, 2017.

[65] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. N. Levine. A secure, efficient, and transparent network architecture for Bitcoin. Technical report, UMass Amherst, Tech. Rep. UM-CS-2016-006, 2016, 2016.

[66] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[67] F. Reid and M. Harrigan. An analysis of anonymity in the Bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[68] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99–100. IEEE, 2001.

[69] J. B. Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the society for industrial and applied mathematics*, 8(1):181–217, 1960.

[70] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[71] J. Schnelli. Peer authentication (bip 150). https://github.com/bitcoin/bips/blob/master/bip-0150.mediawiki.

[72] J. Schnelli. Peer-to-peer communication encryption (bip 151). https://github.com/bitcoin/bips/blob/master/bip-0151.mediawiki.

[73] E. G. Sirer. Bitcoin guarantees strong, not eventual, consistency. http://hackingdistributed.com/2016/03/01/bitcoin-guarantees-strong-not-eventual-consistency/.

[74] Y. Sompolinsky and A. Zohar. Bitcoin's security model revisited. *arXiv preprint arXiv:1605.09193*, 2016.

[75] C. Troncoso, G. Danezis, M. Isaakidis, and H. Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *arXiv preprint arXiv:1704.08065*, 2017.

[76] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.

[77] S. B. Venkatakrishnan, G. Fanti, and P. Viswanath. Dandelion: Redesigning the Bitcoin network for anonymity. *arXiv preprint arXiv:1701.04439*, 2017.

[78] L. Wang and I. Pustogarov. Towards better understanding of Bitcoin unreachable peers. *arXiv preprint arXiv:1709.06837*, 2017.

[79] R. Wattenhofer. *The Science of the Blockchain*. CreateSpace Independent Publishing Platform, 2016.

[80] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[81] J. Wu, S. Guo, H. Huang, W. Liu, and Y. Xiang. Information and communications technologies for sustainable development goals: State-

of-the-art, needs and perspectives. *IEEE Communications Surveys & Tutorials*, 2018.

[82] K. Wüst and A. Gervais. Ethereum eclipse attacks. Technical report, ETH Zurich, 2016.

[83] K. Wüst and A. Gervais. Do you need a blockchain? *IACR Cryptology ePrint Archive*, 2017:375, 2017.