Blockchain-based Sybil Attack Mitigation: A Case Study of the I2P Network

Kotaiba Alachkar & Dirk Gaastra
{kotaiba.alachkar, dirk.gaastra}@os3.nl
MSc Security and Network Engineering
Faculty of Physics, Mathematics and Informatics
University of Amsterdam

August 22, 2018

The disadvantage of fully peer-to-peer systems is their vulnerability to Sybil attacks. In a Sybil attack a single adversary manages to influence the whole system through multiple identities. I2P is a peer-to-peer anonymous communication network that has implemented certain measures to reduce the effectiveness of Sybil attacks on the network. Previous research has provided suggestions for improving I2P's security against such attacks, however, these suggestions do not have the desired impact. Our research provides an in-depth analysis of existing and proposed solutions for the mitigation of Sybil attacks on the I2P network. Our analysis demonstrates that the current state of I2P still leaves the network vulnerable to attacks. Furthermore, proposed solutions provided by earlier research do not succeed in fully removing the attack vectors. Our solution involves using Blockchain as a distributed, tamper-proof medium to keep track of Floodfill routers in the network. We demonstrate that by incorporating Blockchain, the Sybil attack is made infeasible for an adversary. Based on the data collected in the chain, nodes in the network are able to individually determine the overall trustworthiness of a Floodfill router and can decide whether to interact with these routers.

1 Introduction

The increasing amount of online activities over the last few decades has lead to a growing concern for people's privacy and anonymity. Consequently, Internet users have increasingly started using Anonymous Communication Networks (ACNs) to secure their online privacy to varying degrees. Such measures allow users to hide their identity and the nature of their communication, in addition to encrypting the content itself. Providing perfectly anonymous communication over the Internet is important, yet difficult to achieve. Among other ACNs, I2P, the Invisible Internet Project, aims to offer this high level of invisibility and provide anonymity.

Contrary to the well-known and researched TOR project [1], the number of active users of I2P is manageable and analysis of its security rare. As the official I2P network's statistic source, Stats.i2p reports that the overall network consists of approximately 60,000 nodes at any given time. Therefore, the size of the network is relatively small. However, the user-base is increasing significantly with over 3,000 new routers registering in July alone [2]. The fact that I2P runs in a completely distributed fashion has essential advantages. These include better scalability and no trusted central party. However, there are also security risks associated with such a set-up [3]. Malicious peers behaving badly in the absence of an authority can be damaging for the network. For instance, by forging multiple identities to deceive the normal peers, one can control the network partially or in its entirety.

The lack of an authentication mechanism due to the trade-off with anonymity results in a weak defence against Sybil attacks. Although several countermeasures have been proposed by participants [4]. There is no research that verifies these findings, and clear answers do not exist as to which approach is better. In this paper, we will address both retroactive and proactive approaches for the mitigation of Sybil attacks. We will explore how Blockchain techniques can be used to mitigate this attack along protecting the core concepts of I2P: perfect anonymity within a peer-to-peer environment. The paper is structured in the following manner: in Section 2, we will briefly go over some background information which is necessary to put the research into context. This section discusses both the background of I2P and presents an overview of Blockchain technologies. After this, in Section 3, related work into the general mitigation of Sybil attacks is discussed. In Section 4 we will go into the existing mitigation tactics that are already implemented in I2P. Then, we discuss previous proposals to mitigate attacks. Lastly, we propose our own solution. Section 5 is the discussion of our findings. Opportunities for future work are discussed in Section 6. The research is brought to a conclusion in Section 7.

2 Background

2.1 I2P

I2P is a low-latency ACN with built in applications such as Email, IRC, web browsing, and file sharing [5]. A node in the network can either be a server that hosts a darknet service, or a client who accesses said servers to use their services. Connectivity is carried out through tunnels that are constructed using other peers. These tunnels are packet-based and use onion-routing to provide anonymity. The encryption methods used in these tunnels are similar to TOR. Nodes can either communicate with either *NTCP* or *SSU*. *NTCP* is comparable to *TCP* while *SSU* is similar to *UDP*.

The difference with other ACNs, such as TOR, is that I2P is more decentralized. One of the advantages of being decentralized is that it offers the network the potential to gain higher speeds, whereas a more centralized network would experience increased strain on the central components as the network grows [6], [7]. However, one of the major drawbacks of a distributed network is the inherent lack of trust, yet necessary reliance on peers. To get a better understanding of this problem, this section will address the workings of the relevant components of the network followed by a discussion of mitigation of Sybil attacks. Note that less relevant aspects of the network are ignored or significantly generalized. For more information on the I2P network, we refer the reader to [3], [5], [6], [8].

2.1.1 Network Database

The Network Database (netDb) is a distributed database that is used by nodes in the network to retrieve information about peers. There are two types of data maintained in the netDb; *RouterInfos* and *LeaseSets*. The *RouterInfo* is used to contact other nodes in the network. It contains the data described in Table 1.

Table 1: *RouterInfo data [4]*

Component	Description
Router identity	Encryption key, signing key, and certificate
Contact address	Protocol, IP, Port
Publish date	Publish date of router
Options	E.g. Floodfill, reachable, bandwidth
Signature	Signature of the above, generated by signing key

The *LeaseSet* specifies tunnel entry point to reach an endpoint. This specifies the routers that can directly contact the desired destination. It contains the data described in Table 2.

Table 2: *LeaseSet data [4]*

Component	Description
Tunnel gateway router	Given by specifying its identity
Tunnel ID	Tunnel used to send messages
Tunnel expiration	When the tunnel will expire
Destination itself	Similar to router identity
Signature	Used to verify the LeaseSet

2.1.2 Floodfill Routers

The netDb is retrieved from special routers on the network called *Floodfill routers*. Participation in the Floodfill pool can either be automatic or manual. Automatic participation occurs whenever the number of Floodfill routers drops below a certain threshold, which is currently 6% of all nodes in the network [4]. When this happens, a node is selected to participate as a Floodfill router based on criteria such as uptime and bandwidth. It should be noted that about 95% of the Floodfill routers are automatic [3].

The netDb is stored in a Distributed Hash Table (DHT) format within the Floodfill routers [9]. When a resource is requested by a client, the key for that resource is requested from the Floodfill router considered closest to that key. To have a higher success rate on a lookup, the client is able to iteratively look up the key. This means that the lookup continues with the next-closest peer should the initial lookup request fail. The closeness is purely mathematical, so things such as

IP are not taken into consideration, and is determined by the Kademlia closeness metric [4], [9].

2.1.3 Sybil Attack

A Sybil attack [10] on the network aims to compute multiple identities for Floodfill routers so that the attacker's routers are always closest to a particular resource. This could, for example, lead to a Denial of Service (DoS) on a particular resource [4]. The computation, or generation, of an identity is about as simple as generating key pairs. This is a trivial task for modern hardware. Figure 1 illustrates a Sybil attack. Here, the Sybil nodes are closest to the normal node on the left. Therefore, communication between normal nodes is influenced by the malicious Sybil nodes. In order to exhaust the I2P query limit, it is necessary that 8 malicious nodes are near the victim. This is because a node will try a maximum of 8 Floodfill routers that are closest to the desired resource. If an attacker simply wants to log lookups for a resource, one node is enough [3]. Both of the aforementioned scenarios are categorized as a partial keyspace Sybil attack and pose the largest threat to the current network. The goal of a full keyspace attack is to control the entirety of the network. This attack is less feasible with the current network size. Additionally, it becomes increasingly difficult to execute the attack as the network size grows [11]. The Sybil attack can be used as a stepping stone to other attacks. For instance, an Eclipse attack can be executed when enough identities around a node are controlled by the attacker. An eclipse attack is used to isolate communication with the normal node by blocking peer information [3].

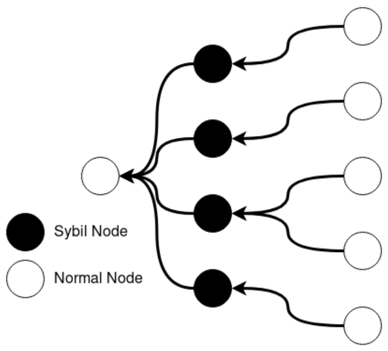

Figure 1: *Sybil attack*

I2P already has countermeasures in place to try to mitigate this attack. One of the measures is the use of two keys for Floodfill routers: an identity key that remains unchanged and a routing key that is used to index a record in the netDb [12]. This routing key is generated by hashing the ID (identity key) and date together. This means that the position of Floodfill routers, as determined by the Kademlia closeness metric, will change every day at midnight (UTC).

2.2 Blockchain Technology

Blockchain is a distributed and decentralized ledger system [13]. It is a 'chain' of blocks, as its name would suggest. A block is an aggregated set of data. Data are collected and processed to fit in a block through a process called mining. Each block could be identified using a cryptographic hash. The formed block will contain a hash of the previous block, so that blocks can form a chain from the first block to the newly formed block. In this way, all the data could be connected via a linked list structure [14].

There are a number of ways to reach consensus in decentralized systems. Most prominent is the use the Proof-of-Work (PoW) and Proof-of-Stake (PoS) algorithms to reach consensus [15]. The purpose of achieving a consensus is to verify that information being added to the ledger is valid. This ensures that the next block being added represents the most current transactions on the network, preventing double spending and other invalid data from being appended to the Blockchain [16]. In addition, the consensus mechanism keeps the network from being derailed through constant forking.

2.2.1 Proof-of-Work Consensus

PoW describes a system that requires a significant but feasible amount of effort in order to deter the exhausting of a computer system's resources by sending multiple fake requests that can result in denial of service. Creating the PoW for achieving consensus is easy to verify but difficult to produce, as it is costly and time-consuming [17].

It comes in the form of an answer to a mathematical problem, which satisfies certain requirements, one that requires considerable work to arrive at, but is easily verified to be correct once the solution has been reached. Producing a PoW is usually a random process with low probability so that, on average, much trial and error is required before a valid PoW is generated [18]. PoW is a requirement to define an expensive computer calculation, also called mining. It needs to be performed in order to create a new block on the Blockchain. A reward is given to the first miner who solves each block's problem [19].

2.2.2 Proof-of-Stake Consensus

PoS is a category of consensus algorithms for public Blockchains that depend on a minor's economic stake in the network [20]. It is a proposed alternative to PoW.

Like PoW, PoS attempts to provide consensus [21], but the process to achieve this goal is different. In PoS-based public Blockchains, a set of minors take turns proposing and voting on the next block, and the weight of each minor's vote depends on its wealth, also defined as stake. Unlike PoW, there is no block reward in the form of newly minted coins. Therefore, the miners typically only acquire only the transaction fees [19].

3 Related Work

Nowadays, privacy and anonymity are gaining an increased amount of attention in both industrial and academic communities [8], [22]. As one of the most used anonymity networks, the I2P network is also receiving wider academic attention. Over the years, many studies have been carried out regarding the mechanism and attacks on the I2P network. The details of the different types of Sybil attacks have been explained in Section 2.2. Therefore, in this section, we will present an overview of the work which attempts to analyze or solve the Sybil attacks. In a variation on resource testing, Awerbuch et al. [23] suggest the use of Turing tests, such as *CAPTCHAs*, to impose recurring fees. This approach is effective for a full keyspace Sybil attack, as many identities need to be put on the network at the same time. However, for a partial keyspace Sybil attack, only a few identities need to be added to the network. Solving a few of *CAPTCHAs* is trivial.

Dragovic et al. [24] suggest requiring certifications for identities, but this certification is not trusted; rather, it is seen as a way of imposing identity creation costs. However, in I2P, only a few Sybil identities are required for an effective attack. Additionally, computational power is tested that mostly involves a one-time cost. Hence, even a high initial cost of claiming a large number of identities could be recovered over time. Regarding resource testing, Cornelli et al. [25] and Freedman et al. [26] have proposed methods to determine if a number of identities possess fewer resources than would be expected if they were independent. These methods include checks for network ability, computing ability, and storage, as well as restrictions on the number of IP addresses. Both papers specifically focused on IP addresses in different domains or autonomous systems. Requiring different IP addresses prevents some attacks but does not discourage others and limits the usability of an application.

Christopher et al. [27] introduced a novel routing protocol for DHTs that is efficient and strongly resistant to the Sybil attack. This so-called *Whanau* uses social connections between nodes to build routing tables that enable Sybil-resistant lookups. As we noted above, there are a variety of solutions that can limit or prevent the attack in several individual application domains. However, while the same concepts can be applied to ACNs, none of the discussed research focused on ACNs, and more specifically, on I2P. This research aims to provide an in-depth analysis of Sybil attack mitigation strategies as they relate to the I2P network. Moreover, we propose both retroactive and proactive countermeasures. The main component of these countermeasures will involve Blockchain.

4 Sybil Attack Mitigation

In this research, we will focus on evaluating existing techniques for the mitigation of Sybil attacks. Furthermore, we will specify our own solutions, based on previous research and ideas. These can be categorized as being either retroactive or proactive. Retroactive solutions are applicable after an attack has occurred whereas proactive solutions aid in the prevention of the attack. We further specify that a proposed solution should satisfy some requirements. A solution should not damage the anonymity of the network as a whole. Furthermore, as I2P is built on peer-to-peer principles, the solution should be as distributed as possible. In addition, to prepare the network for future growth, the solution should be scalable in implementation. Lastly, the solution must be consistent with the network infrastructure as a whole, so that it will not expose the network to other attacks. We will first go over the current state of the network. After that, we discuss the already proposed ideas to mitigate the Sybil attack. Based on the current state and previous research, we conclude the section with our own solution.

4.1 Current State of the I2P Network

The following sub-sections will address the current state of the network, release 0.9.34, announced on April 4th, 2018 [28], coupled with a brief assessment about the used methods.

4.1.1 Router Resources and Parameters

The I2P network already possesses some defense mechanisms aimed at minimizing the risk and impact of Sybil attacks. Running multiple I2P instances on the same hardware could lead to decreased performance of each identity when they are run on inexpensive hardware. Thus, nodes evaluate the performance of known peers and weigh them when selecting peers to interact with instead of using a random selection. As a result, the same host running multiple identities decreases the performance of each of those instances. The number of additional identities running in parallel is effectively limited by the need to provide each of them with enough resources for being considered as peers [3]. However, it should be noted that running multiple identities on modern hardware is very doable.

Furthermore, Sybil attacks can be potent for an adversary when a Floodfill router's identity is free. The primary technique to address this problem is to make identity non-free. I2P developers suggested using HashCash [29] to charge for creating a new identity [11]. However, there is no particular technique implemented yet, but they included placeholder certificates in the router's and destination's data structures which can contain a HashCash certificate of appropriate value when necessary [11]. More information on the proposed PoW solution is provided in Section 4.2.

4.1.2 Keyspace Rotation

The netDb uses a simple Kademlia-style *XOR* metric to determine closeness. The hash of the key being looked up or stored is *XOR-ed* with the hash of the router in question, which results in an integer value that determine closeness. To increase the cost of Sybil attacks, a modification to this algorithm is done. Instead of the hash of the key being looked up or stored, the hash is taken of the 32-byte binary search key appended with the UTC date represented as an 8-byte *ASCII* string *yyyyMMdd* [30]. This is called the Routing Key, and it changes every day at midnight UTC. Nodes clustered at a certain point in the keyspace on one day will therefore be distributed randomly on any other day. The daily transformation of the DHT is called Keyspace Rotation, although it is not strictly a rotation. However, this change does not include any random inputs, and is thus completely predictable [3]. Therefore, an attacker is able to pre-compute the hash values for the next day.

4.1.3 Blacklist

The network will avoid peers operating at IP addresses listed in a blacklist which is managed and published by the people behind I2P. Several blacklists are commonly available in standard formats, listing anti-P2P organizations, potential state-level adversaries, and others. Currently, a default blacklist is distributed with the software, listing only the IPs of past sources of attacks. There is no automatic update mechanism. Hence, should a particular IP range implement serious attacks on the I2P network, contributors would have to ask people to update their blacklist manually through out-of-band mechanisms such as forums, blogs, etc [11]. Another downside is that the blacklist management is rather centralized, giving more power to a single authority.

4.2 Previous Proposed Solutions

This section discusses solutions proposed in previous research. The discussion is followed by an analysis of these solutions, as these solutions are the foundations upon which our solution is built.

4.2.1 Removed Threshold

A proposed solution by Egger et al. [3] involves the removing of a threshold for the number of Floodfill routers. In their research, they argue that due to the current algorithm, a new Floodfill router is only selected once the number of current Floodfill routers drops below a certain number. This construction allows an adversary who is carrying out a Sybil attack to sequentially DoS the Floodfill routers near a resource. Once the Floodfill router being attacked stops being a Floodfill router due to lack of resources, the attacker's identity can get in. Following its entry, the attacker can stop the DoS of that router and move on to the next one.

By removing the threshold, a Floodfill router can temporarily leave and come back again once its resource capacity has returned to normal. This makes an attack less feasible since all Floodfill routers would need to be DoS'ed at the same time for the duration of the Sybil attack. This solution is relatively easy to implement. However, it does not completely solve the problem as an attacker could, instead of DoS'ing routers, generate identities which are closer to the victim.

4.2.2 Floodfill Router Election

Another solution by Egger et al. [3] is to regulate the number of Floodfill routers automatically. This means that one would no longer be able to manually configure their router to become a Floodfill router but would instead be automatically selected based on known metrics. This would mean that the attacker would need to invest more resources so that their router becomes the next elected router. Alternatively, a weighted approach can be taken with a random component so that even a router with the most resources is not guaranteed to be the next in line. The weakness of this solution is that it only partly solves the problem. This is because even if all routers are automatic, it does not mean that they are all trustworthy.

4.2.3 Router Proof-of-Work

As discussed previously, the developers of I2P have been considering to apply a PoW algorithm that can be implemented in the currently unused data structures within a router, namely the certificate field [11]. The PoW algorithm considered is known as HashCash [31], However, the developers have identified two major problems with it. One is maintaining backward compatibility with older versions of the I2P software. Second, the process is not a one-time investment. It needs to be done for every identity that want to join the network. Therefore, pertains to finding an appropriate difficulty level for the PoW so that an attack becomes less feasible but devices with less computing power (such as phones) are still able to create an identity within a reasonable time [11].

4.2.4 Age-based Reputation

An improvement suggested by Egger et al [3] discusses the possibility of having an age-based reputation. In such a scenario, a Floodfill router would only be trusted after a node has directly observed it being active in the network for at least N days. This makes a Sybil attack less feasible as identities need to be regenerated every day, as discussed in Section 4.1. The downside of this solution is that bootstrapping is not considered. If a node is starting up, it will have no information on the age of its peers, as the node needs to make observations on node presence itself.

4.2.5 Trust-based Reputation

In his research, Douceur coined the idea of relying on the collective trust to verify an identity [10]. This solution is similar to PGP, where a web of trust is created to verify someone's identity [32]. The research concludes that such a system will only be another victim for the Sybil attack. With the generation of an arbitrary number of identities, the identities would be able to verify each other. Therefore, such a system will not work without a central authority.

4.2.6 Blacklist Improvements

Improvements to the blacklist can be made by having a subscription-based model, as proposed by the developers of I2P [4]. This blacklist would be controlled by a group of nodes that receive consensus on misbehaving nodes. Alternatively, the list could be in a central location. However, in both these scenarios, the network can be crippled by either a group of nodes working together or by performing a central resource attack.

4.3 Blockchain Implementation

In this section we propose our solution that will facilitate the mitigation of Sybil attacks. The shortcomings of the current state of the network and weaknesses of the previously proposed solutions will be also addressed. Blockchain will be a way for nodes to achieve consensus on identifying a Floodfill router and knowing for how long that entity has been a Floodfill router. Furthermore, the use of Blockchain can result in both proactive and retroactive responses to adversaries.

Firstly, a comparison is drawn between different technologies, which would also involve a defence for our choice, Blockchain, which is a public permissionless ledger. Secondly, we will provide the specifics of the application of Blockchain for our scenario. Moreover, an improvement to the existing keyspace rotation is proposed. Thereafter, the specifics of the block data structure are discussed. Lastly, we discuss the scalability of our solution.

4.3.1 Alternative Technologies Comparison

There exist 2 feasible alternatives to the Blockchain: Tangle and Hashgraph. We briefly explain the concepts of both of these and make a comparison with Blockchain so that it may offer a discussion and defence for our choice of Blockchain.

Hashgraph [33] is based on the notion of having a distributed ledger, like Blockchain. However, it offers an alternative consensus mechanism. It uses techniques like Gossip about Gossip (GaG) and Virtual Voting (VV) to achieve fast and secure consensus. Using GaG, additional information is added to the hashes of the last two people talked to. This means that a node will choose another node at random, and then they will provide each other with all of the information in their knowledge so far. The process will be repeated with a different and random node, and all other nodes do the same. In this way, if a single node becomes aware of new information, it will spread exponentially fast through the network until every node is aware of it [33]. Figure 2 illustrates the history of any gossip protocol represented as a directed graph. Each vertex in the row represents a gossip event. For example, the event at the far right in the node C row represents node A performing a gossip sync with node C in which node A sent all the information that it knows. Hence, right vertices represent earlier events in history when time flows to the left of the graph. The edges between the vertices represent information flow. The edges are bidirectional since information will flow both ways.

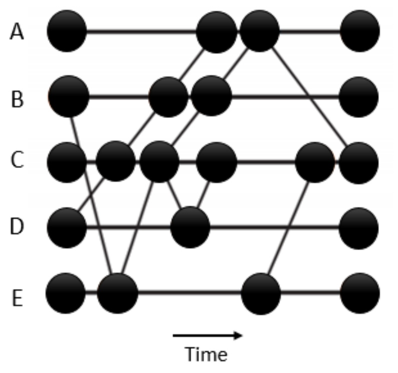

Figure 2: *Directed graph for gossip history*

The additional information, together with the hashes, can then be used to build a Hashgraph that is updated when more information is gossiped to each node [34]. Once the Hashgraph is completed and distributed across the nodes, the system knows what a node would vote, since it is aware of information that each node has and when they knew it. In Virtual Voting, as the name implies, the voting process is virtual This means that no actual votes need to be submitted. The virtual voting is done by analyzing the Hashgraph and determining what a node would have voted [33]. Basically, all nodes have all the information and can thus predict what other nodes would have voted based on the information that is available to these other nodes.

So far, Hashgraph has only been deployed in private, permissioned networks, i.e., as a private distributed ledger [35]. If implemented in a public setting, Hashgraph may face similar challenges as other public distributed ledger technologies (DLTs), especially when it comes to security and performance [36]. Furthermore, Hashgraph currently scales only in the number of transactions processed but does not scale with the number of nodes in the network [35], as it depends on each node having the complete knowledge about the state of the network. More importantly, I2P is a public-permissionless network, the nodes participating in the network are not known beforehand and untrusted since any node is allowed to join or leave the network. Hence, using Hashgraph is not feasible. Moreover, Hashgraph has yet to release concrete technical details for its deployment as a public ledger and it is still closed source [35].

Tangle [37] is a third-generation DLT, based on a Directed Acyclic Graph (DAG): a data structure that moves in one direction without looping back onto itself. The acyclicity is an important concept of Tangle, as it means that not all nodes need to be in sync with each other on the current information [36]. Unlike Blockchain, Tangle does not use blocks. Each transaction performed has to validate at least two previous transactions. This would mean that with each transaction that is added to the 'tip' of the graph, two other transactions from the previous tip are randomly assigned to be validated. By validating a transaction, the validity of the transaction itself as well as conformance to protocol rules is checked [36]. Validating the transaction is done through constructing the bundle and signing of Inputs, tip selection, and PoW. After all these three steps are completed, you can broadcast the transactions to the neighbors and wait for it to be accepted by the network. This process is illustrated in Figure 3.

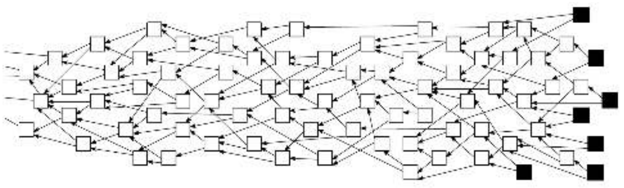

Figure 3: *Tangle incoming transaction flow. White squares represent verified sites, while black squares represent tips [37]*

Such a solution cannot safely be deployed to reach consensus on the I2P network. An adversary will be able to create an arbitrary number of identities that can validate each other. Furthermore, a new node in the network cannot be reasonably expected to be able to safely verify existing peers. Additionally, synchronizing the state between nodes is an issue for existing Tangle implementations. Nodes need not be in sync with each other in Tangle. However, in I2P, all nodes need to be able to verify any Floodfill router. Lastly, for Tangle to become fully decentralized, and thus not rely on a central coordinator, the number of 'transactions' needs to achieve critical mass [38]. The critical mass is accomplished when the network has enough nodes to become self sustaining. This means that when Tangle is implemented, a central authority is necessary, which contradicts the key concept behind the I2P network.

The evaluation summary is offered in Table 3. The choice for Blockchain might not be apparent from this table. However, each feature should be weighed differently. The most important of these features is a public ledger type. Based on the ledger type, some consensus algorithms will not be possible. Moreover, a public ledger is permissionless as well. Another noteworthy detail is that I2P, being an anonymity network, any accompanying technology should also be anonymous as well. The achievement of consensus is less important, as long as the consensus technology is reliable.

Table 3: *DLTs comparison summary [39]*

	Blockchain	**Tangle**	**Hashgraph**
Data structure	Blockchain	DAG	DAG
Ledger type	Public	Public	Private
Permissioned	No	No	Yes
Anonymous	Yes	Yes	No
Consensus	PoW, PoS	PoW	GaG, VV
Efficiency	Low	High	High
Central Authority	No	Yes	No
Copyright	Open-source	Open-source	Proprietary

The peer-to-peer nature of the I2P network suggests that there should be a technology that does not require a central authority. Furthermore, the technology has to be open source due to the fact that adjustments need to be made to fit within the I2P ecosystem. Taking all these points into account, we propose that Blockchain will provide us with the proper foundation to build our solution on.

4.3.2 General Structure

By using Blockchain, some issues of the current state of the network as well as the shortcomings of previously proposed solutions can be addressed. The content of a block in the Blockchain will provide information on the active Floodfill routers. This information includes, but is not limited to, the IP, the router keys, and whether the router was elected automatically or manually. Whenever a new Floodfill router announces itself, it is added to the current block. This way, the age of a router can be verified by anyone at any time by simply traversing the Blockchain. This eliminates the bootstrapping issue for routers, which prevented new routers from verifying the age of their peers. The downside of having an update approach in the implementation instead of having a complete list of nodes with every block is that a node will need to traverse the entire chain to verify a node. However, a complete list in every block would increase the size of the chain to the point of unwieldiness.

Figure 4 provides a high-level overview of the implementation of Blockchain for I2P. The first block of the chain will be a complete list of Floodfill routers that are present at that time. Each subsequent block will essentially be an update to that list. For clarity, transactions that are there to maintain the Blockchain, such as reward transactions for miners, are omitted in the figure. One of the criteria is that the solution should be scalable. Therefore, we consider PoS rather than PoW for the consensus of the Blockchain. This is due to the fact that research has shown this method to be more scalable [40], [41]. Furthermore, PoS allows for low-resource devices to contribute to the chain and ultimately offer a more distributed solution. It should be noted that the downside of a PoS system over a PoW system is that PoS Blockchains can be manipulated by big stakeholders.

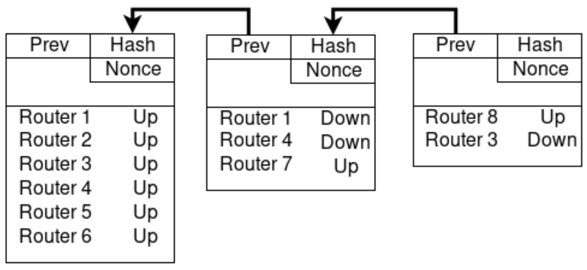

Figure 4: *High-level overview of Blockchain implementation*

PoS relies on a random value, or Nonce, which is generated by the miners of block N to determine the miners of block $N + 1$. The value is generated as a collaborative effort, to avoid a single actor being able to tamper with the random value to ensure their position as miner for the next block. This makes use of a technique called *threshold signature* [42]. As the application of the Blockchain pertains to keeping track of Floodfill routers instead of the classic use case where it is used as a public ledger for transaction, some problems arise. The main issue is the seeming lack of incentive for miners of new blocks. These miners will be the participants in the Blockchain who add new blocks, and are therefore essential. The lack of incentive also makes it difficult for miners to prove their stake. This issue can be solved in different ways.

The first approach is to require a fee each time someone wants to initiate a Floodfill router. This fee is then used to reward the miners for adding new blocks. A benefit of this approach is the added deterrent for attackers to create multiple Floodfill routers. The fee could be incurred every time a new identity is computed and mixed in with the Kademlia hash that will eventually determine the position in the network. The largest disadvantage is that a fee may discourage participation of legitimate Floodfill routers. The second approach is using the Blockchain for facilitating transactions among users of the network. This would essentially introduce a new currency. Using this currency, transaction fees could be paid for every transaction made by a user. This ensures that Floodfill routers participating in the network do not have to pay fees to the miners. However, there would be no incentive for miners to include new Floodfill routers in the block. The miners would simply need to add the monetary transactions for them to reap the transaction fees. This would therefore curb the original intention of the introduction of the Blockchain.

Lastly, new 'tokens' could be minted for every block that is added. This is usually not done in PoS algorithms but could be applied here. However, to avoid devaluation of the currency, it should preferably not be transferable to money, as this would likely give an ever-decreasing monetary value to the tokens. Instead, reputation could be the currency that is awarded for every block. This lays the foundation for a reputation system wherein a node can accumulate reputation by becoming a miner. This reputation can then be used to further verify a node. For instance, there could be a threshold reputation value for Floodfill routers. If a node does not have the required amount of reputation, it cannot participate as a Floodfill router. If a node wants to become a Floodfill router, they have to participate as a miner in the Blockchain first. This makes it so that the Blockchain becomes a system for Floodfill routers, by Floodfill routers. In turn, the Floodfill routers stake their reputation to discourage tampering with the blocks. Taking into account the aforementioned, we propose a reputation-based system.

4.3.3 Improved Keyspace Rotation

To determine a Floodfill router's 'position' in the network, recall that currently, a hash of the router ID together with the date is used to determine its closeness to a certain resource. By using Blockchain, we can take advantage of a non-deterministic keyspace rotation. Instead of creating a hash with the router ID and date to determine a Floodfill router's position in the network, the randomness of the Blockchain can be used. As mentioned before, a random value is computed by a group of miners for every block. This would traditionally be used to help determine the next miners. However, in addition to that, we can use that value to generate the new Kademlia hashes for the Floodfill routers. This can be used instead of using the current method, which hashes the router ID with the current date. For instance, after every 100 blocks, the new positions have to be generated by hashing the router ID with the previous block's random value. This thwarts an attacker's precompute capabilities that are available in the current construction. The exact interval of the keyspace rotation is left up for discussion. The performance impact of this specific improvement on the network is minimal. However, it should be stated that this version of a keyspace rotation requires that all Floodfill routers should be synced with the chain at the same time, which could become an issue in high-latency scenarios.

4.3.4 Block Data Structure

The data structure of the block header can mostly be based on the existing PoS Blockchains. For instance, Figure 5 illustrates the block header structure of `NEO`, one of the leading PoS currencies [43]. `NEO` is similar to `Ethereum` in the sense that it supports smart contracts

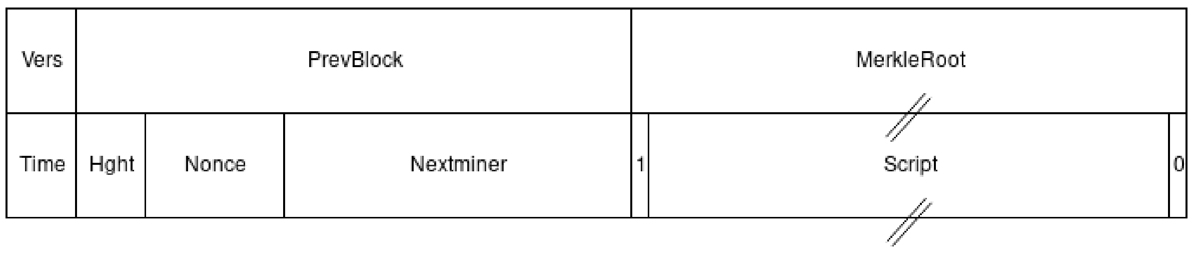

Figure 5: *Block header*

[44], [45]. However, NEO uses PoS while Ethereum still uses PoW. The documentation of NEO's structure is very complete and offers a useful basis for our PoS Blockchain. The block header is entirely designed by NEO, as are the *MinerTransaction* and *EnrollmentTransaction*. The rest of the Blockchain implementation and design choices are our own. In figure 5, the *Vers* field is the version of the block. This field is 4 bytes in length. Currently this is set to 0. The *PrevBlock* field provides the hash of the previous block. It is 32 bytes in length. *MerkleRoot* provides the root hash of a transaction list. This field is also 32 bytes in length. The *Time* field provides a 4-byte time stamp value. The height of the block is provided by the 4-byte *Hght* field. The *Nonce* has an 8-byte field and is basically a random number generated by the miners of a block. Next is the contract address of the next miner, which is given in the *Nextminer* field (20 bytes). This is determined based on the random values given by all miners of a block. If the miner is down, a next miner is determined based on that same number. More information on this is available in [42], [44]. After this is a 1-byte field that is always set to 1. The next field is the *Script* field, which is a field of arbitrary length that holds a script to validate the block. The header ends with a 1-byte field that is always set to 0. After this follows an array of transactions [44]. The hash value of a block is calculated by hashing the first seven fields of the block, which is everything except the script. This includes the *MerkleRoot* of the transactions in the block so any tampering with transactions will not be possible. After the block header come transactions. As a basis, we will keep using NEO [43]. However, changes will need to be made so that it works for I2P. Table 4 offers an overview of all transactions.

MinerTransaction is a transaction that will award the miner of a block to receive a reward. This is the first transaction of each block. The transaction contains output for the miner. In our case, this output will be reputation for the miner. This transaction has a 32-bit random number to add extra randomness to the block as well as avoid hash collisions [44]. Miners get rewards in the form of reputation by validating blocks and their transactions.

EnrollmentTransaction is used by a node to enroll as a miner. This includes the public key of the minor. By signing up, a deposit is sent to the address of the public key. The registration is canceled when this deposit is spent. The node would already require some reputation for this in order to make the deposit. Therefore, one could start with a minimum amount of reputation. This is non-transferable and thus restricts an attacker from registering multiple nodes and then transferring all reputation to one node [44]. The deposit ensures that if a miner behaves maliciously, they lose their deposit. This system lies at the heart of a PoS Blockchain.

RouterUp* and *RouterDown are the main components of the block. When a Floodfill router comes up, it will be added to the block. When this router becomes unresponsive or signifies its departure, a *RouterDown* transaction is done. Having these transactions will make the Blockchain more manageable in size, and ultimately more scalable as only updates have to be added to the chain instead of entire lists of current Floodfill participants. The entire chain is started with a block that includes *RouterUp* announcements for every Floodfill router. Both the *RouterUp* and *RouterDown* transactions include the Router ID (its key), a bit to signify whether the node was automatically selected to be a Floodfill router or if this was done manually.

Table 4: *Transaction types*

Value	Name	Description
0x00	MinerTransaction	Miner reward
0x01	EnrollTransaction	Enrollment for minor
0x13	RouterUp	Announcement of new FF router
0x14	RouterDown	FF router no longer responding

This bit is set to 1 for automatic, 0 for manual. Further, a PoW field is added, which is a submission of the PoW done at the current keyspace rotation. The actual implementation of this is optional, but the field is there should the developers of I2P still want to implement PoW. Lastly, a nonce is added to the transaction to avoid hash collisions. The transactions can be verified by all contributors to the Blockchain by means of pinging a router. If a router is non-responsive, it means that it is down. If it is and was not seen in the last block, a *RouterUp* transaction is valid. Furthermore, a *RouterDown* transaction should be added by a miner instead of the router itself, as an unexpected router crash will not allow for the adding of a new transaction by that router. In addition to the Type and transaction-specific fields, there are some other fields that are shared by all transactions. For instance, every transaction has inputs and outputs. The inputs of *RouterUp* and *RouterDown* transactions are arbitrary, as the transaction itself will hold all information necessary.

4.3.5 Detecting Suspicious Behavior

Using a tamper-proof technology such as Blockchain allows for the recording of suspicious behavior. Therefore, the Blockchain can be used both retroactively and proactively. For instance, when there is a sudden influx of Floodfill routers in a particular area of the network, this could indicate an attack and nodes can make a proactive effort to mitigate such an attack. This can be done on an individual basis by each node specifying their own parameters of what it considers suspicious behavior. Then, it can attempt to avoid nodes showcasing this behavior. By using the Blockchain retroactively, a reference to the Blockchain could substantiate claims made by users. If a user suggests their node was under attack, a reference can be made to certain blocks. Here, suspicious behavior of the attacking identities could easily be verified. Therefore, this can aid the retroactive process of adding IPs to a blacklist.

4.3.6 Blockchain Scalability

To properly investigate the scalability of our solution, we first need to determine the number of nodes over time so that network growth can be taken into consideration. An illustration of this is presented in Figure 6. This graph illustrates a rough estimation of the number of active nodes in the I2P network over time. This information is based on the estimation provided by stats.i2p [2]. The Blockchain will only keep track of Floodfill routers, meaning, not all nodes will be tracked in the chain. This limits the chain size and limits privacy concerns as well. As an estimate, the current rules for automatic opt-in dictate that approximately 6% of the routers in the network should be Floodfill routers [4]. However, should the Blockchain become too large to keep in memory due to transactions accumulating over time, the chain should be reduced in size. This should be done incidentally and in coordination with the community. A possible solution to a large chain is specify a point where the chain will be cut off and a new chain is started. Alternatively, an automatic chain cut could be implemented. However, it is hard to determine in this phase of research whether this is the preferred option. Alternative options should be weighed as cutting the chain will have downsides since an assumption has to be made that the previous part of the chain is trustworthy.

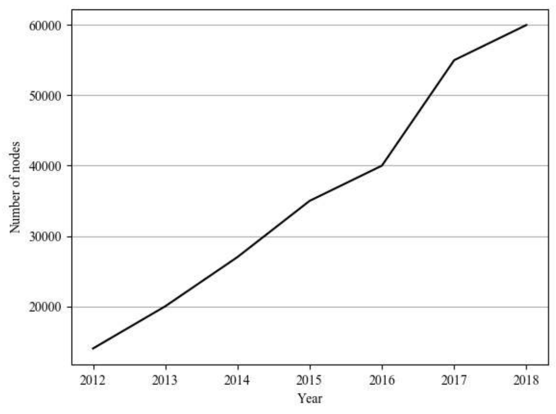

Figure 6: *Number of nodes in the I2P network by year [2]*

An attacker could try to attack the Blockchain by generating a great number of routers and thus filling up the blocks. However, this is mitigated by the fact that a node needs to contribute to the chain and build up reputation before being able to join as a Floodfill router.

5 Discussion

By evaluating the current state of the I2P network, we found issues in the current mitigation tactics against Sybil attacks. The most notable problems were insufficient randomness for keyspace rotation and the relative ease of identity generation. The previously proposed solutions have offered mitigation tactics for these problems. However, these solutions are non-satisfactory due to the existence of security gaps or and the fact that these solutions are incoherent with core I2P principles. Our proposed solution addresses most of the issues. Using Blockchain, a Floodfill router can build up reputation and each node can independently verify said reputation of Floodfill routers. Furthermore, the Blockchain offers a non-deterministic approach to rotating the keyspace. Still, it is not impossible to carry out a Sybil attack. An adversary with enough computing power can still potentially generate numerous new identities. However, this scenario is highly unlikely as the keyspace rotation cannot be determined within reasonable time beforehand. Moreover, if an identity has not been in the Blockchain for long enough, it cannot be trusted as a Floodfill router. The biggest downside of the chosen implementation is the increase in size the chain will endure whenever a router joins or becomes unresponsive. However, weighing the different implementations, we are confident that the proposed design will limit the chain size. Should a reduction in size be necessary at some point, some methods like cutting the chain are discussed. It should also be pointed out that there are some privacy implications that come with our solution. Floodfill routers are usually known. However, keeping track of them publicly might not be a desirable situation.

6 Future Work

Some implementation details have been left unaddressed in this research. Before being able to implement Blockchain as the standard way to mitigate Sybil attacks on the I2P network, more research needs to go into the methods of implementation. For instance, the exact PoS algorithm used to validate blocks still remains unaddressed. Furthermore, the performance of the I2P network is still lacking at this point. Adding Blockchain could place a toll on the existing nodes, which will result in decreased overall performance of the network. The methods proposed in this research have not yet been tested for validity. A comprehensive analysis needs to be made so that security and privacy of I2P users are proven to be safeguarded. Furthermore, this research only focused on the theoretical analysis of other DLTs. A pragmatic approach could be taken in which the functionality is compared in a practical manner. The use of Blockchain in the I2P network could be further explored. For instance, if all information about the network is submitted to the Blockchain, Floodfill routers themselves will become unnecessary. However, this would potentially come at the cost of privacy.

7 Conclusion

Our analysis found that by using Blockchain, Sybil attack vectors can be minimized. A Blockchain can be employed to provide a tamper-proof, distributed medium that keeps track of the activity of Floodfill routers. This way, each node can individually determine the trustworthiness of a Floodfill router by, among other things, determining its age in the network. Using Blockchain adds positive externalities in addition to its main purpose. The Blockchain allows for randomness in the keyspace rotation. By hashing the blocks together with the router identity, the location in the network, as provided by the Kademlia closeness metric, can be determined in a non-deterministic way. For scalability purposes, the Blockchain uses a PoS algorithm. This means that the Blockchain can be maintained by all nodes in the network, regardless of available computing power. By participating as a miner, nodes can build up reputation, which can be a requirement for becoming a Floodfill router. This not only encourages contributions to the Blockchain but also makes Sybil attacks costlier.

8 Acknowledgements

We would like to extend our gratitude to our supervisor Vincent Van Mieghem from Deloitte Netherlands. His insight and extensive assistance during our research has been invaluable. We also thank all reviewers for sharing with us their valuable feedback on earlier versions of the paper.

Bibliography

[1] The Onion Router (TOR) project, official website, 2018. [Online]. Available: https://www.torproject.org/.

[2] *I2p network dashboard*. [Online]. Available: http://stats.i2p/ (visited on 06/11/2018).

[3] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, "Practical attacks against the i2p network", in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2013, pp. 432–451.

[4] *Threat analysis*. [Online]. Available: https://geti2p.net/en/docs/how/network-database (visited on 06/08/2018).

[5] *I2p front page*. [Online]. Available: https://geti2p.net (visited on 06/07/2018).

[6] B. Conrad and F. Shirazi, "A survey on tor and i2p", in *Ninth International Conference on Internet Monitoring and Protection (ICIMP2014)*, 2014, pp. 22–28.

[7] M. Ehlert, "I2p usability vs. tor usability a bandwidth and latency comparison", in *Seminar Report, Humboldt University of Berlin*, 2011, pp. 129–134.

[8] J. P. Timpanaro, C. Isabelle, and F. Olivier, "Monitoring the i2p network", PhD thesis, INRIA, 2011.

[9] Q. Li, H. Li, Z. Wen, and P. Yuan, "Research on the p2p sybil attack and the detection mechanism", in *Software Engineering and Service Science (ICSESS), 2017 8th IEEE International Conference on*, IEEE, 2017, pp. 668–671.

[10] J. R. Douceur, "The sybil attack", in *International workshop on peer-to-peer systems*, Springer, 2002, pp. 251–260.

[11] T. I. official website, *I2p's threat model*, 2010. [Online]. Available: https://geti2p.net/en/docs/how/threat-model.

[12] P. Liu, L. Wang, Q. Tan, Q. Li, X. Wang, and J. Shi, "Empirical measurement and analysis of i2p routers.", *JNW*, vol. 9, no. 9, pp. 2269–2278, 2014.

[13] J. Bagley, *What is blockchain technology? a step-by-step guide for beginners*. [Online]. Available: https://blockgeeks.com/guides/what-is-blockchain-technology/ (visited on 06/08/2018).

[14] *Blockchain - technical details*. [Online]. Available: http://www.doc.ic.ac.uk/~ma7614/topics_website/tech.html (visited on 06/15/2018).

[15] *An introduction to consensus algorithms: Proof of stake and proof of work*. [Online]. Available: https://cryptocurrencyhub.io/an-introduction-to-consensus-algorithms-proof-of-stake-and-proof-of-work-cd0e1e6baf52 (visited on 06/15/2018).

[16] T. Schumann, *Consensus mechanisms explained: Pow vs. pos*. [Online]. Available: https://hackernoon.com/consensus-mechanisms-explained-pow-vs-pos-89951c66ae10 (visited on 06/08/2018).

[17] *Proof of work*. [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_work (visited on 06/15/2018).

[18] *What is proof of work? (pow)*. [Online]. Available: https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/proof-of-work (visited on 06/15/2018).

[19] *Proof of work vs proof of stake: Basic mining guide*. [Online]. Available: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/ (visited on 06/16/2018).

[20] *Proof of stake faq*. [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ (visited on 06/15/2018).

[21] *Proof of stake*. [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_Stake (visited on 06/15/2018).

[22] P. H. O'Neill, *Tor and the rise of anonymity networks*. [Online]. Available: https://www.dailydot.com/debug/tor-freenet-i2p-anonymous-network/ (visited on 06/03/2018).

[23] B. Awerbuch and C. Scheideler, "Group spreading: A protocol for provably secure distributed name service", in *International Colloquium on Automata, Languages, and Programming*, Springer, 2004, pp. 183–195.

[24] B. Dragovic, E. Kotsovinos, S. Hand, and P. R. Pietzuch, "Xenotrust: Event-based distributed trust management", in *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, IEEE, 2003, pp. 410–414.

[25] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Implementing a reputation-aware gnutella servent", in *International Conference on Research in Networking*, Springer, 2002, pp. 321–334.

[26] M. Freedman, "A peer-to-peer anonymizing network layer", PhD thesis, Massachusetts Institute of Technology, 2002.

[27] C. Lesniewski-Laas and M. F. Kaashoek, "Whanau: A sybil-proof distributed hash table", 2010. [Online]. Available: https://pdos.csail.mit.edu/papers/whanau-nsdi10.pdf (visited on 06/25/2018).

[28] *0.9.34 release - the i2p official website blog*. [Online]. Available: https://geti2p.net/en/blog/post/2018/04/10/0.9.34-Release (visited on 06/12/2018).

[29] *Hashcash*. [Online]. Available: http://www.hashcash.org/ (visited on 06/12/2018).

[30] *Kademlia closeness metric*. [Online]. Available: https://geti2p.net/en/docs/how/network-database#kad (visited on 06/12/2018).

[31] A. Back *et al.*, "Hashcash-a denial of service counter-measure", 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf (visited on 06/12/2018).

[32] P. R. Zimmermann, *The official PGP user's guide*. MIT press, 1995.

[33] L. Baird, "Hashgraph consensus: Fair, fast, byzantine fault tolerance", Swirlds Tech Report, Tech. Rep., 2016. [Online]. Available: http://www.swirlds.com/wp-content/uploads/2016/06/2016-05-31-Swirlds-Consensus-Algorithm-TR-2016-01.pdf (visited on 06/25/2018).

[34] S. Technologies, *Hashgraph blockchain: Similarities differences*. [Online]. Available: https://www.sofocle.com/hashgraph-blockchain-similarities-differences/ (visited on 06/18/2018).

[35] Y. Jia, *Demystifying hashgraph: Benefits and challenges*. [Online]. Available: https://hackernoon.com/demystifying-hashgraph-benefits-and-challenges-d605e5c0cee5 (visited on 06/18/2018).

[36] P. Schueffel, "Alternative distributed ledger technologies blockchain vs. tangle vs. hashgraph-a high-level overview and comparison", 2017. [Online]. Available: https://www.researchgate.net/publication/323965938/.

[37] S. Popov, "The tangle", *cit. on*, p. 131, 2016.

[38] S. Technologies, *Iota tangle takes the blocks out of peer-to-peer, distributed ledger networks*. [Online]. Available: http://microgridmedia.com/iota-tangle-takes-blocks-peer-peer-distributed-ledger-networks/ (visited on 06/18/2018).

[39] P. Schueffel, *10 years blockchain. the race is on: Blockchain vs. tangle vs. hashgraph*. [Online]. Available: http://fintechnews.sg/16989/blockchain/10-years-blockchain-the-race-is-on-blockchain-vs-tangle-vs-hashgraph/ (visited on 06/19/2018).

[40] J. Spasovski and P. Eklund, "Proof of stake blockchain: Performance and scalability for groupware communications", in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, 2017.

[41] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, *et al.*, "On scaling decentralized blockchains", in *International Conference on Financial Cryptography and Data Security*, Springer, 2016, pp. 106–125.

[42] V. Daza, J. Herranz, and G. Sáez, "Some applications of threshold signature schemes to distributed protocols", 2002. [Online]. Available: https://eprint.iacr.org/2002/081.pdf (visited on 06/25/2018).

[43] T. Mo and D. t. Pangwae, 2016. [Online]. Available: https://github.com/neo-project/neo/wiki/Network-Protocol.

[44] *Neo white paper*, 2016. [Online]. Available: http://docs.neo.org/en-us/.

[45] V. Buterin *et al.*, "Ethereum white paper", *GitHub repository*, 2013.