Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Kim Aksel Tahuil Borgen

A study on committee-based sharding within the context of Rapidchain

Master's thesis in Computer science
Supervisor: Mariusz Nowostawski
July 2020

NTNU
Norwegian University of Science and Technology

Kim Aksel Tahuil Borgen

A study on committee-based sharding within the context of Rapidchain

Master's thesis in Computer science
Supervisor: Mariusz Nowostawski
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Committee based sharding presents an interesting solution to the scalability problem of traditional Nakamoto-based blockchains because the throughput in these systems scales linearly with respect to the total amount of nodes in the system. The state of the art solution within the field of committee based sharding is at present the Rapidchain protocol presented by Zamani *et al.* [1] who present performance comparable to traditional systems, such as Visa and Twitter. The key questions that this thesis asks are: Are there any limitations or other issues in the protocol presented by Zamani *et al.* [1], can these matters be addressed or resolved, and can the results be replicated? This thesis implements a derivation of the implementation presented by Zamani *et al.* [1]. The problem of cross-committee transaction verificability which is not presented in the existing literature is discussed and a solution to this problem is provided by using the novel contribution of ‚Proof of Consensus. Due to limitations in this thesis, the results cannot conclusively compare the presented results by Zamani *et al.* [1], but similar trends are discovered.

Sammendrag

Kommite basert "sharding" presenterer en interesant løsning til skalabilitetsproblemet av tradisjonelle Nakamoto baserte blockchains fordi "throughput" i disse systemene skalerer linært i forhold til den totale mengden antall noder i systemet. Den beste nåverende løsningen innenfor komite baserte "sharding" er per i dag Rapidchain, presentert av Zamani *et al.* [1] som presenterer resultater som kan sammenliges med tradisjonelle systemer slik som Visa og Twitter. Hovedspørsmålene som denne oppgaven presenterer er: Finnes det noen begrensninger eller andre problemer i protokollen presentert av Zamani *et al.* [1], kan disse begrensningene bli løst, og kan resultatene bli reprodusert? Oppgaven implementerer en derivasjon av implentasjonen presentert av Zamani *et al.* [1]. Problemet med å verifisere transaksjoner på tvers av kommiteer, som ikke er presentert av dagens literatur, er diskutert og en løsning ti dene problemet er utarbeidet ved å bruke en bevis på konsensus. Dette er det orginale bidraget i denne oppgaven, på grunn av begrensninger i denne oppgaven, det er ikke mulig med sikkerhet å sammenlige resultatene, men lignende trender er blitt observert.

Contents

Figures

Tables

Chapter 1

Introduction

Bitcoin, a construct introduced in 2008 by Nakamoto [3], solved the double spending problem of existing similar solutions with a construct called Proof of Work (PoW). This construct enabled the creation of an immutable blockchain, where a distributed network of trust-less nodes agree, or more correctly, reaches consensus on a common block.

The key issue in current blockchain systems is scalability [2], slow throughput and large latencies. In fact, Bitcoin's maximum throughput is 7 transactions per second, with close to an hour average latency before a transaction is included in a block [2]. Several modern systems have produced better scalability, but any scalability observed in traditional blockchains is pale in comparison with real-life traditional systems, such as VISA, with their 2000-47000 transactions per second, or Twitter with their 5000 transactions per seconds [22].

My previous work [2] asked the key question of how academia are addressing the scalability issue of blockchain technologies. The work resulted in several possible areas of research, where one of the most important areas was the area of committee based sharding. The concept of dividing the blockchain into multiple smaller blockchains produced excellent performance results. The latest of these solutions was identified to be Rapidchain by Zamani *et al.* [1] which produced state of the art results of a throughput of 7300 transactions per second, with a user-perceived latency of 70.7 seconds with a network size of 4000 nodes. My previous work did not however not find any studies that thoroughly investigated Rapdichain so the key questions that this study asks are: Are there any limitations or other issues in the protocol presented by Zamani *et al.* [1], can these matters be addressed or resolved, and can the results be replicated?

The main contributions of this work is as follows.

- Presents a literature review of every study that references Rapidchain.
- Clarifies, defines and addresses required missing implementation details or issues into a rework of the original implementation by Zamani *et al.* [1].

- Implements a derivation of the Rapidchain protocol. The results of this implementation is compared to the original to determine to what degree the results presented by Zamani *et al.* [1] can be reproduced.
- Presents the novel problem of cross-committee transaction verificability.
- Presents the novel contribution Proof of Consensus that addresses the verificability problem. The Proof of Consensus construction enables the construction of a proof to verify the inclusion of any transaction in any block in any committee.

The composition of the thesis is as follows. Chapter 2 goes into the background of Rapidchain. Chapter 3 presents the literature review. Chapter 4 presents a rework of the Rapidchain protocol. Chapter 5 presents the experimental and analytical results of the implementation of the rework presented in Chapter 4. Chapter 6 discusses the results and presents the limitations of this study. Chapter 7 concludes the study and presents identified future work.

> **Remark.** In the rest of this thesis the words "original", "Rapidchain", and "authors" usually refers to the study by Zamani *et al.* [1] that introduced the Rapidchain protocol, unless it is clear that something else is referred to.

Chapter 2

Background

This section presents the background for the work in this thesis. Related works are excluded since they are naturally included in the literature review presented in Chapter 3.

2.1 General

Proof of Work (PoW) is a way for a node to prove that they have used energy to work on a specific problem. Bitcoin uses the computational problem of computing a hash of a block, with a varying nonce, lower than a specific threshold. PoW consensus is also called Nakamoto consensus [1, 21]. Proof of Stake (PoS), as presented by King and Nadal [29] replaces the cost of producing a Proof of Work with a direct monetary cost.

The problem of a group of distributed, trust-less, nodes collectively agreeing to a value can be reformulated as the problem of reaching consensus on a value. This was formalized into the byzantine generals problem, presented by Lamport *et al.* [30], and is solvable only if the number of faulty nodes is below 1/3.

From the byzantine generals problem, one can define a byzantine system to be a system of nodes where some are faulty. Subsequently, the property Byzantine Fault Tolerance indicates that a system of nodes tolerates up to 1/3 faulty nodes [2].

A faulty node, within the context of Byzantine environments, is a node which does not behave as expected. A faulty node can be both honest and malicious. If a node is honest, then a fault may happen due to unexpected circumstances, such as package loss or power failures. But if a node is malicious it may choose to fault and produce any behaviour possible, for example voting for an invalid value.

The scalability trilemma, further defined in Borgen [2], states that between the

three properties scalability, security and decentralization, only two of the properties can be achieved at the same time.

In a unspent transaction output (UTXO) system, a transaction consists of several outputs and inputs. An input of a transaction must be an output of a previous transaction that has not already been used as input in any confirmed transaction. Bitcoin also makes the optimization of requiring that the full value of the output must be included as the input in a new transaction. In the literature, the nomenclature of spending an UTXO refers to the process of including the full UTXO in an input in a new transaction.

Confirmation latency is defined by Zamani *et al.* [1] to be the time between a transaction being included in a proposed block to the time the block is finalized and therefore immutably added to the blockchain. The user-perceived latency is defined to be the time between when a transaction is sent to the system, until the transaction is included in a finalized block.

Hash trees, as presented by Merkle [13], more commonly refereed to by the nomenclature Merkle trees [1, 4], enables the construction of a hash tree over a range of data. The root of this hash tree naturally encapsulated the identifier of the data, since any changes to underlying data will result in a different root. This property is particularly useful in blockchain systems. Where the root of the Merkle tree computed over the data of a large block, it can be used as the block's ID. To create a chain of these blocks, it is enough to add the previous ID of a block to the data of the next block. Any block can then be verified against the indicated root in this chain of IDs, and that is how we get the word blockchain.

An equally important property of a Merkle tree is the ability to create a proof for any of the leaf nodes in the Merkle tree. Given the root of the Merkle tree, a single path of nodes in the hash tree, and the leaf node itself (or the hash of it), one can verify that the leaf node exists in the Merkle tree with the given root. The single path of nodes in the hash tree is defined to be the Merkle proof. This is usefull if a Merkle tree is constructed over the set of transactions included in a block. If the root of the transaction set is included in the block header. You can then verify that a transaction is included in a block by the block header itself, and through a Merkle proof of the transaction.

Sharding, within the field of blockchain, can be defined to refer to the set of solutions that split either nodes or data into smaller subsets of the full blockchain, while at the same time achieving some kind of inter-operability between shards [2].

2.2 Committee based sharding

Committee based sharding differs from normal sharding in that both node and data are divided into shards. Each subset of nodes will then have exactly one data shard. These subsets of nodes form a committee. These committees create

their own internal blockchain, and each committee performs a consensus process between the committee members to agree on the state of its internal blockchain. All the committees in a system that agree on the protocol used, form a system of multiple smaller blockchains. In theory, the system should then have increased the maximum possible throughput by number of committees. In practice, this perfect scalability cannot exist since the committees must also communicate together.

The scalability factor, with respect to committee based sharding, can therefore be defined as the amount of increased throughput when adding either a singe node or committee. Rapidchain observes this factor, with respect to single nodes, to be linearly increasing [1].

To observe a scalability effect, a transaction submitted to the system should only be handled by a subset of committees. If every committee must process a transaction, then no performance increase should be observed. In Rapidchain, a single transaction only has one responsible committee. A problem arises when considering an UTXO system in a committee- based sharding scheme. If the process of determining the responsible committee of a transaction is random, then the inputs of a transaction may belong in several different committees. The corresponding outputs in the different committees either have to be destroyed or moved to the responsible committee before the transaction can be confirmed. If the corresponding outputs in the different committees were not provably spent, the issue of double spending becomes a problem.

Due to the problem presented in the previous paragraph, a protocol to handle cross committee outputs must be implemented. This thesis defines the nomenclature of such a protocol to be a "*cross committee transaction protocol*". Different nomenclature is used in literature, for example "*cross-shard transactions*" is used by Zamani *et al.* [1].

2.3 Rapidchain

Only the relevant components of Rapidchain that are referenced in this thesis are presented here. For a more complete overview, the reader is strongly recommended to read the original study by Zamani *et al.* [1].

2.3.1 Information Dispersal Algorithm (IDA) gossiping

Zamani *et al.* [1] presents a gossiping protocol for gossiping large messages within committees. The authors state that the gossiping protocol was inspired by the IDA protocol of Alon *et al.* [31], and they therefore call their protocol IDA Gossip. The protocol divides a message M into equal-sized κ chunks. Several parity chunks Φ are added to provide redundancy using the error correcting code protocol of Reed and Solomon [32]. To reconstruct the original message it is enough to have any κ amount of chunks in the concatenated list of chunks $\kappa||\Phi$. A Merkle tree is then

computed over the concatenated list. The concatenated list is then divided equally to the neighbouring nodes. Each message to the neighbouring nodes includes the root of the Merkle tree, Merkle proofs of the chunks, and the chunks themselves. Each neighbour will then gossip the received messages to its neighbouring set, and so on. If each honest committee member receives at least κ distinct chunks, then the message can be successfully reconstructed.

2.3.2 Consensus protocol

Zamani *et al.* [1] state that they implement a variant of the synchronous consensus protocol of Ren *et al.* [9]. The synchronous assumption state that all messages are received by every member of the committee within a known time Δ. Due to the synchronicity assumption, the total amount of faulty nodes within a single committee can be as high as 49%.

The consensus protocol is run in iterations. A new leader is picked at each iteration. The leader gossips a new proposed block to the committee using the IDA Gossip protocol. The leader then gossips the block header of the proposed block with the tag "propose". Each honest node that receives this message will gossip the block header with the tag "echo". If no malicious behaviour is detected, by for example proposing multiple valid blocks, then all honest nodes gossip the block header with the tag "accept". If a node observes 51%, or $mf + 1$, valid "accepts" it will accept the block and move on to the next iteration.

2.3.3 Cross committee transactions

If a transaction has two inputs I_1 and I_2 and one output O, and the two inputs belong in different committees than the responsible committee for the transaction itself, then two new transactions, with their respective inputs, and their respective outputs being of the same size as the input, are sent to the committees C_{ini} that own the UTXOs referred to in the inputs. This process is accomplished by the leader of the committee C_{out}. C_{ini} then includes the transaction in a new block and spends the UTXO by sending it back to C_{out}. If all new cross committee transactions are successful, then a final transaction spends the outputs of the successful cross committee transactions, into the original output O. If one or more cross committee transactions are not successful, then the UTXO of the successful cross-committee transactions is still available to the user and only the committee ownership of an UTXO has been shifted.

2.3.4 Inter-Committee routing

Zamani *et al.* [1] states that they use ideas from the Kademlia protocol, as presented by Maymounkov and Mazieres [10], to create a committee to committee routing protocol. This protocol produces a routing table with the network information of $\log(\log(n))$ nodes in each of the $\log(n)$ closest committee. Any further definition

of the inter-committee routing protocol was not inherently clear and is therefore further defined in Section 4.2.

Chapter 3

Litrature Review

3.1 Review protocol

As in my previous work Borgen [2], a review protocol is presented to reduce the possibility of researcher bias, and to increase replicability. This review protocol was created before conducting the search. The purpose of this literature review may seem very general, because at the time of conducting this review I did not know precisely which areas to focus on. The purpose of this review was therefore to gain general insight into what literature has already discussed about Rapidchain.

3.1.1 Review questions

- RQ1: Is the Rapidchain protocol state of the art?
- RQ2: Are there any areas of the Rapidchain protocol with shortcommings or limitations that should be investigated?
- RQ3: Have there been any attempts to recreate either the Rapidchain protocol or recreate results of some of the components Rapidchain introduces?
- RQ4: Is there any research in the literature about committee based sharding that can be applied to Rapidchain?

3.1.2 Search Process

As was discovered in the previous work [2], there is a significant amount of relevant grey literature, literature that is not necessarily published but can be classified as scientific research, that was not discovered with the scopus database. Due to this, the search should be carried out in the google scholar search engine [24]. Google scholar will for example present white papers that are not searchable in traditional databases, such as scopus.

Similarly to the previous work [2], the search is conducted in three rounds.

The first round will only include titles and abstracts. The full content of the study should be skimmed to determine inclusion or exclusion only if there is any doubt. If a conclusion cannot be easily determined, then the study should be included in the next round.

All studies should be thoroughly examined in the second round, and a final inclusion or exclusion conclusion must be presented with a minimal justification.

The last, and third round only includes studies that should be presented. These studies will also undergo data extraction and quality assessments.

3.1.3 Search method

Due to the general nature of the review questions specifically on the Rapidchain protocol, a simple search of all the studies that reference Rapidchain should be enough to satisfy review question 1-3. The previous work [2] indicated that Rapidchain was a state of the art committee based sharding solution, with significant improvements over its predecessors. Due to this we can assume that all research on committee based sharding, should reference the Rapidchain protocol, and we can conclude that the presented search method will loosely satisfy review question 4.

3.1.4 Inclusion criteria

- Literature that presents discussions, extensions, or improvements on Rapidchain, committee based sharding, or any relevant component.
- Literature that presents results that may be applicable to Rapidchain, but do not discuss that applicability directly.

3.1.5 Exclusion criteria

Most of the exclusion criteria presented in this section are based upon the exclusions criteria in the review protocol of the previous study [2].

- Literature not applicable to permissionless or public blockchain technology will be excluded.
- Literature that has a small, irrelevant, auxiliary section pertaining to the main topic of the research will be excluded as it does not contribute to the literature.
- Literature focused on economic, business, hardware, or other non relevant aspects will be excluded.
- If the literature has low credibility, as discovered in the quality assessment presented later, it should be thoroughly examined if the results presented are correct. If a conclusion cannot be reached, it should be clearly stated in the results.
- If the literature has severe shortcomings without proper discussion or analysis it should be excluded.

- Literature with non-scientific or equivocated nomenclature or otherwise incomprehensible arguments, should be excluded due to the risk of misunderstanding the content of the literature.
- Literature that utilizes a trust or reputation mechanism should be excluded if and only if it does not present a rigorous and correct security analysis.

3.1.6 Study selection process

I, alone was responsible for the search process.

3.1.7 Quality assessment

The Quality Assessment checklist can be found in table **??**. The checklist is directly modified from the previous work [2], with some non-relevant questions removed. This checklist was initially based on Table 5 and 6 in Keele *et al.* [25].

Each question should be answered with a score between 0 and 10, but if the question is not relevant to the study then it can be skipped.

3.1.8 Data extraction

The data to extract from included studies is defined in Table **??**.

3.1.9 Data analysis

Quantitative data should be extracted, presented and discussed from the overall search process. Quantitative data should also be presented for the last round.

3.1.10 Review timetable

This review was conducted in the spring of 2020.

3.2 Included and excluded studies

The search presented in this literature review was extracted from google scholar on 27. April 2020. The search yielded 155 studies, of which 3 were duplicates. The first round yielded 45 accepted and 107 excluded studies. After round two, a total of 20 studies were selected for inclusion in this literature review, and 20 studies were excluded. Figure 3.1 presents a flowchart of the entire search process.

The quality assessment, and data extraction, as presented in Table **??** and **??** respectively, for each of the twenty final studies, can be found on the accompanying GitHub repository[1].

From the twenty identified studies, only three distinct categories could be identified. Incentive mechanism, security and error correcting codes. Relevant results

[1] `https://github.com/kimborgen/master-thesis-data`

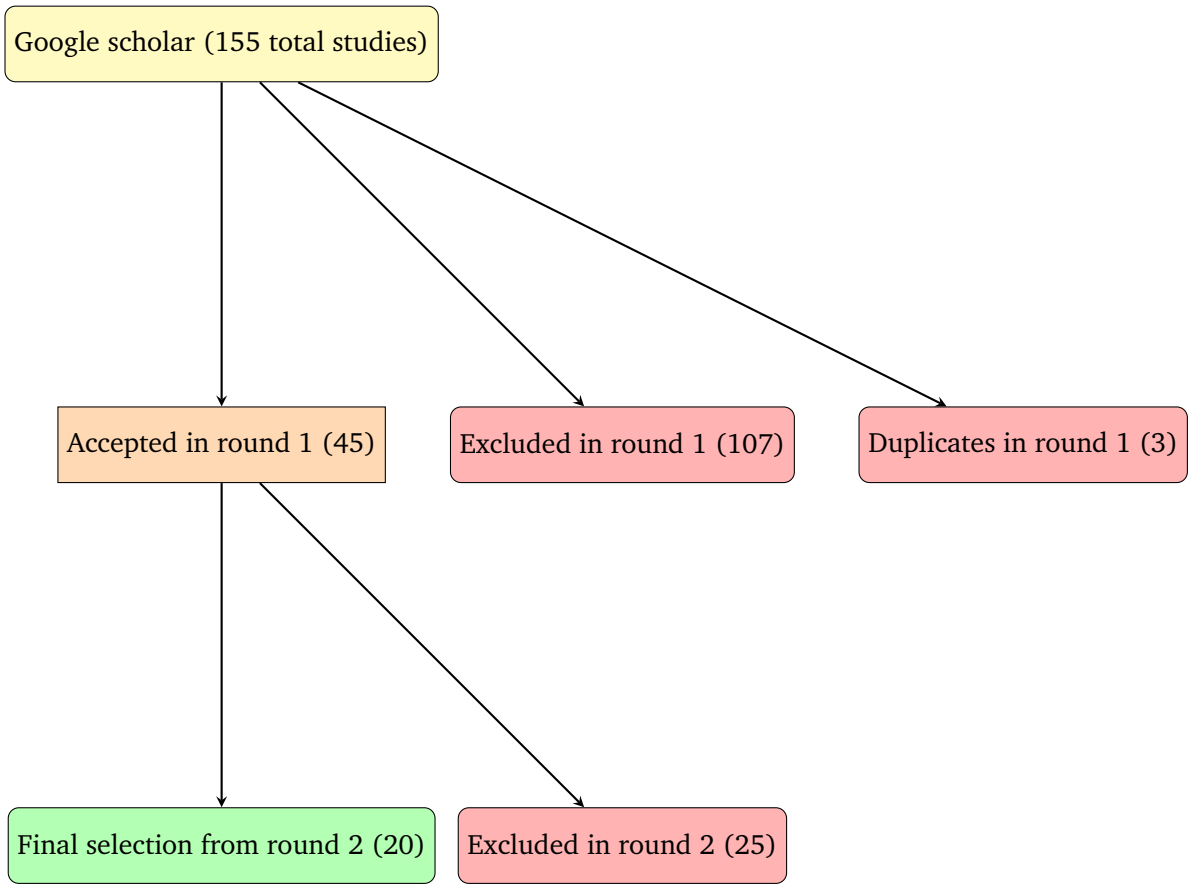


Figure 3.1: Flowchart of the search process

that do not fit under the three identified categories are therefore placed under a fourth general category called miscellaneous. All classifications are presented in Table 3.1.

Classification	Studies
Security	8
Incentive scheme	4
Error correcting codes	4
Checkpoint mechanism	2
Protocols	2
Cryptographic primitives	2
Load balancing	1
Client driven protocols	1
Storage size	1
Cross chain transaction	1
Atomicity	1

Table 3.1: Identified classifications on the twenty included studies

3.3 Literature review results

3.3.1 Incentive mechanism

Manshaei *et al.* [39] presents a model of commitee-based sharding, along with a non-cooperative N-player game model The authors proved that a cooperative equilibrium, which is the state where the protocol is successful, could not be achieved if rewards are uniformly distributed to all participants. The authors then presented fair incentive where participants are only rewarded if they participate in consensus. This mechanism proved that if the set of transactions were large enough for every participant, then cooperative equilibrium could be reached. The authors then presented a novel incentive protocol that relies on an elected coordinator per shard that recommends the optimal strategy for each participating node. The results indicate that less transactions are required to reach cooperative equilibrium in the novel inventive-compatible mechanism. The presented incentive mechanism in Kokoris-kogias [33] is similar to the fair incentive mechanism presented by Manshaei *et al.* [39].

Manshaei *et al.* [39] states that Rapidchain and its predecessors have a "*lack of clarity*" and therefore assumes that the networks fail an epoch if one or more shards fail.

Wang *et al.* [62] show that none of the presented committee based sharding protocols have any incentives for participating in protocol. Indicating a significant area of further research, Bano *et al.* [61] confirms this by stating that there has been little investigation into incentives for committee based protocols.

Wang and Wu [34] cite the work of Manshaei *et al.* [39], but take a different and more "*aggressive*" approach. The authors present an incentive game where a validator stakes the original transaction fee in a transaction as a way of confirming the transaction. Any node can challenge this transaction, either if the outcome of the transaction is false, or if they are acting maliciously, by staking the total amount of transaction fee that is related to that transaction. If the Challenger wins, they receive the last validation stake which doubles their initial stake. This process can proceed in rounds until a threshold is met, and the dispute is taken to a special committee that resolves which side is right. This process lies under the assumption that the special committee is not malicious, and that the threshold is sufficiently low so that honest nodes are able to challenge a wealthy adversary. The results show that a rational and honest participant will have a positive payoff, and participants that guess, abort or act maliciously will have a negative payoff. The paper however scored very low on the "clear and coherent" quality assessment, which means that it is difficult to judge the overall contribution of the paper. The credibility of the results can therefor be questioned. Nevertheless, the resulting concept is novel and interesting and is therefor included in this literature review.

Harmony [36] state that they will use an incentive mechanism of rewarding new tokens, along with the transaction fees, equally to all voting shares that signed the block. This is similar to the fair incentive as explained by Manshaei *et al.* [39]. The authors also state that they will use harsh penalties, by slashing stakes, if any dishonest behaviour is confirmed. However, any details, analysis, discussion or evaluation was not performed on this incentive mechanism.

Chawla *et al.* [65] showed that by using error correcting codes, in the context of a traditional Nakemoto consensus blockchain, it was possible to increase block size. The higher block size incentivizes miners to actually include as many transactions as possible, because the accumulated transaction fee is greater than the reward of mining an empty block. Whether or not this is relevant to a commitee-based protocol is yet to be determined.

3.3.2 Security

Das *et al.* [42] states that the security of the byzantine agreement protocols is strictly correlated with the size of the identity set. Furthermore, the authors state that this sample size is the limiting factor on the identity protocol. The authors then explain that in order to improve communication cost, several protocols implement a trade-off of performance when in the honest case, which lowers security.

Hafid *et al.* [52] bounds the failure probability for both one epoch and one committee using three different probability bounds. The results indicate that Hoeffding's probability bound from Hoeffding [5][2], gave the best approximation. Hoeffding's probability bound can be used to estimate failure probability for both one epoch

[2]not included in this literature review

and one committee for any parameter. This can also be used to calculate the average number of years before failure. This will allow researchers and implementers to choose the level of security by choosing parameters such as committee size. Hafid *et al.* [52] further presented some estimated numbers using this probability bound that are presented in table 3.2.

Committee size	Failure probability	Years to fail
80	<1.3E-03	
150	<1.4E-04	
250	3.79E-07	450
800	2.83E-23	235970

Table 3.2: Estimated failure probability of Rapidchain presented in Hafid *et al.* [52] using a total network size of 4000 participants

Rajab *et al.* [76] analyzed the sybil identity resistance of Elastico, as presented by Luu *et al.* [23][3], to work out the probability of both breaking the consensus process and the probability of controlling the protocol based on the adversaries hashing power. Their results show that an adversary can break the protocol by controlling 25-33% of the hash power.. This makes sense since the total resiliency of Elastico is 1/4.

Homoliak *et al.* [63] presents various attack vectors for various blockchain components. The authors did not analyse Rapidchain, and the section on BFT was severely lacking, nevertheless any implementation of Rapidchain should be analysed within the work of Homoliak *et al.* [63] to ensure that any potential attack vector is discussed.

Wang *et al.* [62] state that Rapidchain does not take into account what happens in a cross-chain transaction scenario where a committee leader is malicious.

Wang *et al.* [62] further state that to prevent prevent adversary attacks, no one should be able to work out the result of the reconfiguration before it happens. They should also not be able to influence the result of the partition in a way that gives them prior knowledge of the results. This is also formalized by Bano *et al.* [61].

Bano *et al.* [61] states that using PoW, or any similar construction, will impose a limitation of sybil resistance because the miners with a higher hashing power will have a bigger likelihood of dominating a committee.

Bano *et al.* [61] further discuss that a leader-based committee protocol introduces several challenges including, Denial-of-Service attacks directly targeted at committee leaders, and malicious leaders detection that degrades performance.

Bano *et al.* [61] also identifies that a single shard failure compromises the security of the entire system.

[3] not included in this literature review

Manuskin *et al.* [53] discusses the vulnerability to Denial-of-Service attacks in sharded systems. The authors argue that since transactions are assigned to shards based on their hash, it is possible for an adversary to create many transactions targeted at a single shard in order to overwhelm the shard.

Harmony [36] state that the Distributed Randomness Generation protocol in Rapidchain is not secure because malicious nodes can send inconsistent shares that create multiple possible versions of the randomness. The authors cite the work of Syta *et al.* [6][4], but do not give any more detailed explanation. They then propose using a Verifiable Delay Function, as proposed by Boneh *et al.* [7][5] and used by the Ethereum 2.0 implementations, combined with a Verifiable Random Function, as the one currently in use by rapidchain, to create an unbiasable and verifiable random number. The authors state that this prevents the last-revealer attack, as explained by Dworzanski [8]. The authors do mention that this approach is vulnerable to situations where an adversary with more computation power than the honest nodes will be able to calculate the randomness slightly before the honest nodes, which could lead to situations where the adversary has an advantage.

3.3.3 Error Correcting Codes

Li *et al.* [41] introduced a coded sharding scheme that mixes several normal data shards into one coded shard the size of one normal shard. This provides data replication, to ensure that data is not lost even if one shard fails, but introduces computational redundancy.

Choi *et al.* [56] comments on the work of Li *et al.* [41] by stating that the high communication burden of using coded data shards limits the application in practice, because required shards must be transmitted to the shard responsible for the resulting transactions.

Choi *et al.* [56] further presents a scheme that uses a network coded practical byzantine fault tolerant algorithm for data replication and communication. This scheme starts with a client sending a transaction to a single node that is responsible for sharing a subset of the data with other backup nodes.

Harmony [36] claims that their solution, which uses RaptorQ's fountain code, is better than Rapidchain's fixed-rate Reed-Solomon codes because such codes may lead to transmission failures once symbols are exhausted. While fountain codes do enable infinite, just-in-time generation of encoding symbols, no more detailed analysis or discussion was provided to back up this claim.

Chawla *et al.* [65] also presents a solution using rateless fountain codes in the context of a normal Nakemoto consensus blockchain. The authors showed that by using error correcting codes it is possible to increase block size while keeping the orphan rate at an acceptable level. This results in higher miner profitability

[4] not included in this literature review
[5] not included in this literature review

and transaction throughput, but negatively effects computational efficiency and latency.

3.3.4 Miscellaneous

Woo *et al.* [72] presents two algorithms dedicated to solving dynamic load balancing in Ethereum's sharding environments. The first algorithm predicts future transaction gas load based on previous gas usage. The second algorithm places the result from the first algorithm into a priority queue, that is used to relocate accounts across shards in order to balance the gas consumption of the system. They experienced up to 12% increase in transaction throughout and up to a 74% decrease in transaction latency. The authors did not discuss Rapidchain in any other capacity than to introduce sharding.

Hafid *et al.* [52] discusses two blockchain projects in industry, Harmony by Harmony [36] and Zilliqa [20][6]. The authors state that both these projects achieve high throughput, but Zilliqa has severe shortcomings because it does not shard state, and the sharding process is vulnerable to singe-shard takeover attacks. Harmony [36] do however claim that they address these problems.

Bano *et al.* [61] state that a client-driven protocol, such as Omnilegder by Kokoris-Kogias *et al.* [21][7], where clients are responsible for part of the cross-committee transaction process, is vulnerable to Denial-of-Service attacks that may result in clients losing all the involved value.

Bartolomey [54] state that Rapidchain has less storage overhead than Omniledger due to omitting any data that contains fully spent UTXO's, rather than implementing a checkpoint mechanism. The author discusses the fact that this assumption is "rather strong" due to the existence of high-stake transactions that optimally require more validation. This trade-off, that does not treat low or high-stake transactions differently, but consequently decreases the security of high-stake transactions, differs from similar protocols. The authors remark that this trade-off, along with its several other improvements, makes Rapidchain the best option for a general purpose blockchain.

Matzutt *et al.* [73] presents a novel pruning method for integration with the existing Bitcoin blockchain, along with a survey of pruning mechanisms. The authors take inspiration from the fast state sync approach of Ethereum that ties the current state to each block. This allows a node to only download the current state with the entire header-chain. The node can then easily verify that the state was valid at that specific block and continue as normal from that block. The authors then present CoinPrune, where snapshots of the current UTXO set, along with other metadata, is made periodically by miners and it's hash is included in those blocks' coinbase (making it available in the header-chain). Each miner that follows the CoinPrune meachinsm will include the snapshot identifier (a hash of the

[6] not included in this literature review

[7] not in this review

snapshot) in their blocks coinbase for a certain reaffirmation time after this. A new node can then bootstrap its blockchain by downloading the entire header-chain, along with a snapshot, and sync as normal after the snapshot. A new node will verify the snapshot by counting snapshot identifiers in the header-chain, and will only accept the snapshot if it has the majority of the reaffirmations. The authors provided only a limited security discussion, but there does not seem to be any large security hole. The authors perceived a storage overhead decrease from 230GiB to 5GiB and synchronization time from 5 hours to 46 minutes for new nodes joining the system. This is because new nodes only need to download the current, pruned, UTXO set.

Wang *et al.* [62] note that the cross-chain mechanism in Rapidchain causes three transactions for every cross-chain transaction, increasing the total number of transactions that need to be processed, subsequently increasing the communication burden. These transactions also depend on the routing algorithm, and the authors state that this is a potential bottleneck.

Dang *et al.* [64] discusses the fact that since Rapidchain does not provide atomicity of cross-shard transactions, it is unsuitable for a non UTXO based system, for example an asset based account system.

Harmony [36] presents a commitee-based sharding blockchain called Harmony, that builds upon Rapidchain and uses many of its components. Harmony breaks the assumption of Rapidchain that all participants have equal computing power, and employs a Proof-of-Stake system where the staked capital determines how many voting shares the participant has in the next epoch. Large stakers would therefore have more control over the system.

Harmony [36] uses both the reconfiguration protocol of Rapidchain, but with voting shares as first-class citizens, as well as the Information Dispersal Algorithm, but with different codes.

Kokoris-kogias *et al.* [47] introduced the allegedly first asynchronous distributed key generation, by creating a novel high-threshold asynchronous verifiable secret sharing construction. These constructions result in a shared key that can encrypt data, generate signatures with constant sizes representing a threshold of participants and otherwise scale blockchains. The authors only mentioned Rapidchain in the context of the latter.

Kokoris-kogias [33] presents MOTOR, a BFT consensus protocol intended to replace the one in Omniledger [21][8]. The author utilizes BLS signatures in an aggregation signature scheme, as presented by Boneh *et al.* [19]. This aggregation scheme transforms multiple signatures into one single verifiable signature, significantly decreasing the computational time to verify blocks. The BLS scheme is also used for threshold cryptography, where a committee can only produce a valid signature if a threshold of nodes signs the message.

[8] not included in this literature review

3.4 Discussion of literature review

3.4.1 Incentive mechanism

The original study by Zamani *et al.* [1] does not present or discuss an incentive scheme, but it is obvious that an incentive scheme should be developed before the application to real world use.

Manshaei *et al.* [39] did not assume malicious adversaries, therefore the derived theorems and equations need to be adjusted to account for the specific threat model in committee-based sharding protocols. The proposed novel incentive mechanism lacks details and discussion. Some potential weaknesses of the mechanism include the need for a coordinator that could produce malicious information, as well as the need for the coordinator to control rewards that could result in unfair exclusion or selfish reward distribution. The number of transactions required per epoch in the proposed mechanism is in the range of 7000-12000, which may affect smaller blockchains. Lastly the model that the authors presented differs from the model of Rapidchain, and the results may therefore not be applicable.

Manshaei *et al.* [39] results are however interesting and could be used to determine if any proposed incentive mechanism would produce a cooperative equilibrium. The naive fair reward concept could also provide a simple incentive mechanism for Rapidchain. The concept of the novel incentive mechanism should be explored further, for example, an interesting question could be how performance would be affected.

It is unclear how the incentive mechanism of Wang and Wu [34] affects performance since the proposed mechanism was also bundled with other novel components outside the scope of this literature review. The evaluation did however give an indication that increases in delay were on the scale of 1.5 -> 4.5 times the original delay depending on the ratio of adversaries, but an estimation of delay was not provided. However one can assume that this incentive approach will increase the communication burden significantly since all rounds need to be broadcast and enough time has to pass for a node to challenge the transaction. In the case of a dispute, the delay of the special committee protocol must also be included. The worst case delay for a transaction may therefor be very large and not feasible.

Wang and Wu [34] designed this mechanism specifically for "*Validation intensive Transactions*", transactions that for example execute smart contract code, and these transactions further use an off-chain computation scheme. So the high delay may be justified in such a system, but since all individual transactions in Rapidchain require the same low validation effort this incentive mechanism might not be suitable.

Harmony [36] introduces the use of penalties to the incentive mechanism. However the use of a penalty system requires something to be staked, and therefore cannot be directly applicable to Rapidchain in its current Proof of Work form.

Chawla *et al.* [65] shows that a transaction fee needs to be large enough such that the reward for including a transaction in a block is higher than by mining an empty block. This is an important property to have in any incentive scheme.

3.4.2 Security

The results of Das *et al.* [42] show that the security of Rapidchain comes directly from how many nodes are present in the system. The larger the network size, the more secure the system is. Rapidchain does not trade-off performance in the honest case.

Hafid *et al.* [52] results shows how much of an improvement Rapidchain is in relation to its academic and industry counterparts, further establishing it as the best solution in the literature at the time. This is mainly because Rapidchain's threat model allows up to 1/3 total adversaries and 1/2 adversaries in each committee, compared to the 1/4 total adversaries with 1/3 committee adversaries in its counterparts. As an example, the authors present a failure probability of 1.34E-02 using a committee size of 250 on the Rapidchain counterparts, with 4.7 days to failure. When compared to the results in table 3.2 Rapidchain has a several orders of magnitude better failure probability for the same committee size.

The results from Rajab *et al.* [76] are not directly applicable to Rapidchain due to the difference in adversary power, but its theorems seems applicable to Rapidchain with modifications. One can therefore use the work of Rajab *et al.* [76] to find out if the hashing power of an adversary breaks the total resiliency of 1/3 in Rapidchain.

The result of Wang *et al.* [62], and subsequently of Bano *et al.* [61], may indicate that the the reconfiguration and distributed random generation protocol should be investigated for possible sources of bias. The inconsistent share problem, as presented by Harmony [36], should also be addressed in a revised distributed random generator protocol. The bias of such a protocol must be thoroughly addressed. In fact, the work of Wang *et al.* [62], Bano *et al.* [61], and Harmony [36] indicate that there might be a bias problem in any distributed random generation protocol. The following research question can be formulated. Is there potential for bias in any component of the Rapidchain protocol?

Rapidchain assumes that every node has equal hashing power, but this is not true in a real life scenario, as described by Bano *et al.* [61]. A big miner can therefore increase the number of included identities it can produce and therefore increase the likelihood of controlling a larger amount of nodes in each committee. This problem is however present in any blockchain protocol.

The possibility to detect a malicious leader, as described by Bano *et al.* [61], should be explored. Is it, for example, possible for a committee to produce a proof of malicious behaviour that can be used to penalize malicious nodes?

Manuskin *et al.* [53] do not discuss the possibility of an incentive mechanism that

would make such attacks costly. Honest transactions could simply raise their transaction fee to be included. However, such Denial-of-Service attacks are prominent in any blockchain system, but since a single shard failure would affect and possibly break the entire system its impact may be more significant in Rapidchain and it should be accounted for in a potential incentive mechanism.

3.4.3 Error Correcting Codes

The concept of storing shard data on several shards will increase security and lower absolute failure probability. Li *et al.* [41] used a relatively low number of nodes in each shard, where the failure probability of one shard is higher. The probability of a shard failure in Rapidchain is only high when the number of committee members are low. A replication scheme may therefore help in small Rapidchains, but it may also provide the ability to recover from shard failures, somethimg which was not discussed in the Rapidchain paper Zamani *et al.* [1].

Choi *et al.* [56] concept of dividing data into several nodes might be usefull if implemented on a committee level, for example by partitioning each checkpoint state into multiple shards, but the method of communication is mainly covered by the IDA protocol of Rapidchain. Choi *et al.* [56] did not discuss their method compared to the conceptually similar IDA protocol.

The results of Chawla *et al.* [65] are not be applicable to Rapidchain due to the fact that Rapidchain does not use Nakemoto consensus, but nevertheless it shows that bigger blocks are propagated faster than by not using error correcting codes.

Neither Harmony [36] or Chawla *et al.* [65] discussed the difference between the fixed-rate code that Rapidchain uses, and the rateless code of fountain codes. Neither did the authors of these two studies discuss fountain codes in the relevant context of Rapidchain. Any relevant performance impact of using fountain codes for Rapidchain can therefore be questioned.

3.4.4 Miscellaneous

Ethereum has an account based system, where one specific shard contains one specific account and all its transaction history. Rapidchain on the other hand uses a UTXO system where each transaction is independent. Due to this fact, the work of Woo *et al.* [72] cannot be applied to Rapidchain, but the observed results may indicate that improving load balancing for an UTXO system might provide better performance.

Tee work of Bano *et al.* [61] shows that since the Rapidchain protocol is not client-driven then a fatal Denial-of-Service attack is mitigated. This indicates the security improvement of the Rapidchain protocol over the Omniledger protocol in Kokoris-Kogias *et al.* [21].

Bartolomey [54] state that Rapidchain does not utilize a Checkpointing mechanism. The authors of Rapidchain did however state that a checkpoint mechanism was utilized, but did not go into any more details. The results of Bartolomey [54] do however indicate that Rapidchain is more secure and better suited for general use than its predecessors.

The work of Matzutt *et al.* [73] is not directly applicable to Rapidchain due to its design being influenced by the need for applicability to the current running Bitcoin blockchain. However, the concept of the checkpoint mechanism of Matzutt *et al.* [73] seems applicable to the Rapidchain protocol.

The results of Wang *et al.* [62] indicate the problem of every transaction potentially increasing the amount of transactions that need to be processed. A transaction can have many cross chain transactions, not just three. Nevertheless the bottlenecks of the cross-committee transaction protocol and the routing protocol should be investigated.

Dang *et al.* [64] state that the design of Rapidchain makes a switch from an UTXO based system to an asset based account system impossible. However, one can design a system where special accounts can be created that belong to the receiver of a transaction if and only if all transactions involved are successful, if not, the output would belong to the original sender of the transaction and no assets would be lost. This approach would incur an extra computation cost since it would require special design. It is unclear as to how this approach would perform under a smart contract scenario, but there are indications that atomicity can be emulated using the current cross-committee design of Rapidchain.

The study by Harmony [36] is classified as a white-paper, an overview of their technical solution intended for industry deployment. As such it did not include any details, comprehensive discussion, analysis or evaluation, but the abstract concepts presented by the authors are relevant. The use of the reconfigruation protool and IDA protocol of Rapidchain in this study Harmony [36] further indicate that the design of these components are state of the art. Indeed, the authors did not find any negative aspects to these approaches.

Rapidchain participants sign blocks individually, and gossip both its own signature and others. This potentially leads to a large storage and communication overhead. A shared key scheme, as presented by Kokoris-kogias *et al.* [47], with constant-sized signatures may therefore provide storage and communication performance increase. The authors however did not evaluate their constructions in terms of performance. Neither did the Rapidchain paper go into depth in relation to the performance impact of their scheme, so it is unclear if there is a performance increase by changing the protocol in favor of a shared key scheme. Such a change would also change many high-level components of the protocol. The asynchronous verifiable sharing construction may be applicable if Rapidchain changes its assumption from synchronous/partly synchronous to asynchronous.

Kokoris-kogias [33] use of a cryptographic aggregation scheme, and threshold

encryption scheme may indicate that the use of a more advanced cryptographic scheme can significantly increase performance. If for example the size of signatures becomes a bottleneck, then an aggregation scheme could be employed to limit the size of the signature set to a single compact signature.

3.4.5 Review questions

This subsection will answer the review questions presented in the review protocol 3.1.

- RQ1: Is the Rapidchain protocol state of the art?
  - None of the studies presented in this chapter indicate that a better solution has been presented in the relevant literature. Harmony [36] does present a solution that claims to be better than Rapidchain, but due to lack of results and analysis, this statement cannot be verified.
- RQ2: Are there any areas of the Rapidchain protocol with shortcomings or limits that should be investigated?
  - The lack of incentive mechanism in any committee-based sharding protocol, as presented by Wang *et al.* [62] and Bano *et al.* [61], indicate a clear area of research.
  - Wang *et al.* [62] indicates a lack of discussion of the consequences of a malicious leader.
  - The results of Wang *et al.* [62], Bano *et al.* [61], and Harmony [36] indicate the possibility of bias in the different protocols.
  - Wang *et al.* [62] also indicate a potential bottleneck in the cross committee transactions protocol
- RQ3: Has there been any attempts to recreate either the Rapidchain protocol or recreate results of some of the components Rapidchain introduces?
  - Hafid *et al.* [52] produced a probability bound for the failure probability of Rapidchain, confirming the failure probability presented by the Rapidchain paper. No other study attempts to reproduce the results of Rapidchain, and no attempt of a second implementation was found.
- RQ4: Is there any research in the literature about committee-based sharding that can be applied to Rapidchain?
  - Most of the studies presented in this chapter had a potential application on Rapidchain. The discussion in Section 3.4 provides the answer to this review question.

Chapter 4

Reworking the Rapidchain protocol

This chapter introduces several shortcomings, drawbacks and missing details of the Rapidchain protocol as presented by the original paper [1]. Some solutions will be presented or discussed.

4.1 Consensus protocol

How does an honest leader re-propose pending blocks, and why does a pending block need to be committed to the blockchain?

There are three scenarios for a non valid block. (1) A leader equivocates by sending multiple proposed blocks or block headers, (2) the block contains invalid transactions or other invalid fields, or (3) it did not receive the required amount of valid signatures within the defined time. If any of these scenarios are fulfilled the block is defined as pending. The pending block cannot be committed to the blockchain since a block needs $mf + 1$ "accept" votes to be valid, and honest nodes do not send "accept" votes on a non-valid block.

The authors did not define the process of re-proposing these blocks referred to in the previous paragraph, and gave no motivation for why a non-valid block should be re-proposed. A pending block should not produce changes to any state, since it is invalid. Therefore there does not seem to be any reason as to why a pending block should be committed to the blockchain. The authors did not describe any block creation protocol. The only reason to commit a pending block would be if the block creation protocol produced state changes. This could happen if for example the block creation protocol only included transactions added to the transaction pool in the previous iteration. Therefor if a pending block was not committed it would cause every transaction committed to the transaction pool in that iteration to be lost. We can however define a block creation protocol that operates on a

independent transaction pool, where transactions are not removed from pool until they are in an accepted block.

A potential motivation for committing pending blocks could be to detect if too many pending blocks were created due to scenario (3) where there is not any dishonest behaviour involved, but rather where the synchronicity assumption might have been broken due to a low Δ. This could produce the necessary data required for a delta-changing protocol or ceremony.

Another potential motivation for committing pending blocks could be to have an irrefutable record of a potential dishonest leader. But because in practice the synchronicity assumption could break, and a pending block is created by scenario (3) without dishonest behaviour involved, then punishment or blacklisting should be done carefully, and preferably in conjunction with an incentive mechanism.

Given the proper motivation for re-proposing a pending block it could be as simple as proposing a chain of blocks. Each pending block could then be wrapped in a null-block, that does not produce any state changes, but encapsulates the original block, so every committee member can confirm that the block is indeed invalid, and check that they have previously denied the block. This null-block would then reference the previous block and have its own unique hash. If there are several pending blocks then the next null block can reference the previous null block, and so on, until a new block is proposed that references the last of the null blocks. This way we will have an uninterrupted hash-chain that can be voted on.

The implementation presented in this thesis does not have a need for the motivations discussed above, and will therefor not be re-proposing blocks. If a block is invalid, or pending, it will not be committed and the next leader will propose a new block. With a different, but most likely similar, transaction set.

How does nodes in a committee communicate messages in the consensus protocol?

The authors only state that each committee member "*gossips*" it's consensus messages. It is unclear if the full IDA-Gossip protocol or normal gossiping to neighbours is utilized. The authors do however state that the they use a variant of the synchronous consensus protocol of Ren *et al.* [9]. The protocol of Ren *et al.* [9] utilizes all-to-all communication.

Due to the fact that the committee sizes are relatively small, and the implementation presented in this thesis is highly parallelizable, the all-to-all communication pattern of Ren *et al.* [9] is utilized in this implementation.

How is the leader election protocol in committees defined?

The authors only state that the current epoch randomness is used to elect the current leader. They do however define a leader election protocol used in the

bootstrapping protocol. Can this protocol be used at the start of each committee iteration?

The authors call this protocol "*Subgroup election*", where each member runs a distributed random generator protocol to generate a random string s. Each member calculates $h = H(s||ID)$ and if $h <= 2^{(256-e)}$, where $e = 2$, is true, it announces itself as leader. Given a synchronous round, then each member in the committee will honor the node with the lowest observed ID.

Since an epoch contains many iterations, this approach would only elect a single leader for every epoch. We mitigate this issue by including the current iteration in the calculation $h = H(s||i||ID)$.

In a committee leader election protocol we can replace s with the epoch randomness, as the authors stated. Since the set of members in a committee is known to every committee member, each node can easily verify if they have the lowest h, by calculating h_{cm} for every committee member, without setting an explicit bound on h. The implementation presented in this thesis assumes that every node remains online, and therefor the node with the lowest h will assume the role of committee leader automatically without exchanging any messages.

> **Definition 1** (Leader election protocol)**.** If s is the current epoch randomness, and assuming every committee member cm, with ID cm_{ID}, is online. At the start of every iteration i, all members will calculate
>
> $$h_{cm} = H(s||i||cm_{ID}) \tag{4.1}$$
>
> for every committee member cm in C, including itself. The node with the lowest h_{cm} will then be accepted automatically by all committee members.

A more advanced leader election protocol must be designed if we break the assumption that every node remains online. Such a protocol can be implemented by choosing a threshold t. If $t = 2$, then if a node calculates that h is either the lowest, or second lowest compared to the set of calculated h_{cm} they can announce their h to every committee member. Given a synchronous round every committee member will now agree on the lowest observed h, and the announced message ensures that the leader is online. But such a protocol is outside the scope of this thesis.

How are iterations synchronized among committee members?

The authors made no statement on when an iteration starts. A simple solution would be to assume a perfectly synchronized clock. Each committee member would then start the leader election protocol in given intervals defined by Δ. However, no perfectly synchronized clock exists in the world. We could approximate such a clock in a controlled testing environment but that this would be impossible in the real world. A better solution would be to start leader election after each

block is accepted and processed into the blockchain. That way we ensure that an iteration is entirely finished before starting a new one. This approach trades off a bit of potential delay with consistent iterations.

> **Definition 2** (Leader election protocol start)**.** The leader election protocol will be initiated by every node right after the previous block is accepted and processed.

4.2 Inter-committee routing with Kademlia

The authors state that they construct their inter-committee routing protocol based on ideas from the Kademlia protocol presented by Maymounkov and Mazieres [10].

What distance metric should be utilized?

The choice of distance metric was not clear. The authors suggested the use of the Hamming distance metric, as presented by Hamming [11], but gave no reason or discussion on the choice of this metric. Maymounkov and Mazieres [10] presents the XOR metric, which is used as the foundation for the rest of the Kademlia protocol. Due to this, the implementation presented in this thesis utilizse the original XOR metric.

Routing protocol definition

It is also unclear how the routing protocol operates and how much of the Kademlia protocol is used. Is the kademlia protocol used only to gather routing information or is it used to physically send messages? The two possible solutions are presented in the following list:

(1) The original Kademlia protocol is only used for routing information. To find a target committee C_t that does not belong in the routing table in the sending committee, then a subset of the nodes in the sending committee C_s will then issue "*find_node*" messages to the committee C_1 in their routing table that is closest to the target committees ID, based on the XOR distance metric. If the target committee belongs in committee C_1's routing table, the information on C_t is sent back to the sending committee C_s. Now C_s has the required network information to send a message to the target committee C_t. If C_1 does not have C_t in it's routing table it will return the committee C_2 that is closest to the target committee, and the sending committee will again send a "*find_node*" message to C_2. This process is performed up to $\log(n)$ times because each committee stores the information of $\log(n)$ committees.

(2) The full message is sent from the sending committee C_s to the first committee C_1 in the routing path. That committee will route the transaction to the

next committee in the routing path C_2, until the transaction is received at the target committee C_t.

If (2) is chosen, it will put a significant burden on the network since the full message needs to be sent to each committee in the routing path. Option (1) is more lightweight since you only have to send the target committee ID (32 bytes in the implementation presented in this thesis), and the response would be the ID and IP pair of a subset of the nodes in the next committee (32 bytes for ID plus 6 bytes for IP and Port per node).

Option (1) is also a subset of the solution presented by Maymounkov and Mazieres [10], whereas option (2) only utilizes the routing table and distance metric of the Kademlia solution.

Due to the reasons presented above, option (1) would be the better choice, and is the one that the implementation in this thesis utilizes.

Is this routing protocol susceptible to Denial of Service attacks?

Due to the fact that transaction, or messages, are routed without confirming the validity of the transactions it may be very easy to perform cheap Denial of Service attacks by flooding the network with invalid transactions that target the longest routing path. Due to insufficient computational resources this could not be investigated in this thesis.

Is a routing protocol necessary?

Given the existence of a cheap Denial of Service attack, it may seem as if the routing protocol trades-off security for scalability.

The authors state that storing the required network information for every node in the system will compromise privacy, simplify Denial-of-Service attacks, and limit scalability. However, each node already stores the ID of each node in the current epoch in the reconfiguration block created by the reference committee. For a network size of 4000 nodes this ID set only takes 128 kilobytes of storage. The required network information of a node can be as low as 6 bytes, 4 bytes for the IP and 2 bytes for the port number. For a network size of 4000 nodes this will only add 24 kilobytes of storage. This information could be sent with the reconfiguration block in each epoch, or a separate network layer could be employed so each node could easily gather the information of the ID's in the reconfiguration block it does not have network information for. This could be done, for example by employing the Kademlia protocol.

Such a system would compromise privacy and simplify Denial of Service attacks somewhat, but every traditional blockchain system already suffers from the same issues.

The implementation in this thesis will not utilize such a system because it is being

used to test the limits of the original Rapidchain protocol, although it may give an indication for future work.

4.3 IDA protocol

How does Rapidchain create a well-connected random graph between committee members?

The authors state that "*Rapidchain creates a well-connected random graph for gossiping between nodes on a committee*", but does not state how Rapidchain creates a well-connected random graph.

If neighbouring committee members are picked at random there is no guarantee that the committee graphs are well-connected. In fact, creating a distributed well-connected random graph is not an easy problem [12], and is outside the scope of this study.

Since all committee members are known to each other it is an easier task to construct a neighbouring set deterministicly based on heuristics that guarantee connectivity. We already have such a construction in the Inter-committee routing protocol, and therefor we employ the same protocol and distance metric to create a routing table of neighbouring nodes. If we ensure that committee ID's and node's ID's are both 32 bytes, the exact same algorithms can be applied.

The choice of the size of the neighbouring set was not presented or discussed by the authors. So to ensure that the gossiping guarantees presented by the authors still hold with this method we could supplement the above routing protocol by for example choosing the nodes with distance $2^i - 2^i/4$ along with the original nodes with distance 2^i until we have d neighbours.

4.4 Cross committee transaction protocol

The cross committee transaction protocol presented in the original paper was very short and incomplete, and the following section is therefore divided into several sub-sections that will clarify and discuss details and inherent limitations of the presented protocol, and then present an improved protocol that addresses these limitations.

4.4.1 Clarifications

How are transaction divided into committees?

The authors state that "*each committee only stores transactions that have the committee ID as their prefix in their IDs*". The XOR distance metric, as presented by Maymounkov and Mazieres [10], can also be seen as a common prefix distance metric, and can therefor be used to fulfill this statement. The committee with

the lowest XOR distance to the transaction ID will be responsible for the transaction.

> **Definition 3** (Committee ownership of transaction)**.** A transaction belongs to the committee ID with the lowest XOR distance to the transaction ID.

4.4.2 Small improvements

Ensure the uniqueness and security of derived cross committee transactions

To ensure that each derived cross committee transaction is valid, and its output can only be sent to C_{out}, then the creator of a transaction must sign every input with the transaction ID.

> **Definition 4** (Signing a transaction)**.** When creating a transaction a user must sign every input with the transaction ID as defined by:

$$\text{Signature}_i = \text{Sign}(\text{Input}_i || t_{ID})$$

Every input can now only be sent to the committee that owns the transaction ID, as defined by definition 3. The output of every cross committee transaction response is also automatically addressed to the owner of the input UTXO. Together they ensure that a cross committee transaction response will only transfer the UTXO to C_{out}, while the ownership of the UTXO stays the same.

4.4.3 Inherent Limitations

What happens if either C_{out} or C_{in} has a malicious leader?

The damage a malicious leader can do in an honest committee is limited to some censorship and bias. A malicious leader can choose which transactions to include in a block, and order those transactions in any order, but a malicious leader cannot create an invalid or inconsistent block. Therefore, a malicious leader can choose to not include any cross committee transactions in its block, but the next honest leader will include it. This will only increase the potential delay, but will still satisfy liveness.

However, since the authors state that only the leader in C_{out} will send the derived cross-committee transactions, a malicious leader can choose to not send these cross-committee transactions and at the same time deceive the rest of the committee that it sent the transactions.

It is therefor clear that the leaders of the committees should not be fully responsible for sending cross committee transactions, or their responses. To solve this issue we can use the same ideas from the inter committee routing protocol, where only $\log(m)$ nodes in a committee will route a transaction.

> **Definition 5** (Responsibility of cross committee transactions)**.** A cross committee transaction, or i's response, is sent by log(m) members of a committee, including the leader.

But how does a leader communicate to the rest of the committee that the original transaction and derived cross committee transaction have been processed and should be sent? And which log(m) nodes should be responsible?

Original, or derived cross committee transactions, are not added to the blockchain in C_{out}

A simple solution to ensure that every original transaction, and its derived cross committee transactions, are sent, is to add it to the current block. A block should be accepted before the derived cross committee transactions are routed because a block may be invalid. After the block is accepted, no leader can re-propose the original transaction. These transactions should not produce any state changes, but rather act as proof of transaction processing.

> **Definition 6** (Proof of Cross Committee Transaction Processing)**.** Every transaction that requires one or more derived cross committee transactions is added to a block, along with its derived cross committee transactions. These transactions are to be specially marked so that they are not processed, and therefor do not produce any state changes.

Once a block is accepted. The log(m) nodes will send the cross committee transactions to their destination committees, but how do we choose these nodes?

Which log(m) **nodes should be responsible for sending the transaction?**

How do we maximise the probability that these log(m) nodes will send the transaction?

To maximize the probability of success we need to pick nodes at random, but only from the set of active nodes. Assuming the existence of a signature set, as defined in definition 11, it can be used to detect which public key was active in that iteration. We can borrow from the leader election protocol and select the log(m) nodes from the last signature set that has the lowest $h = H(s||i||ID)$.

> **Remark.** This will automatically add the leader to the set of nodes since the leader always has the lowest h of the active set of nodes in an iteration.

> **Definition 7** (Responsibility of cross committee transactions)**.** A cross committee transaction, or its response, as presented in definition 6, in an accepted block, will be sent to their target committees by log(m) committee members.
>
> These committee members are chosen from the set of public keys presented in the accepted blocks signature set and is the log(m) nodes with the

lowest $h = H(s||i||ID)$. These $\log(m)$ nodes are called responsible nodes.

Since the set of committee members is known, each member can easily verify if they are responsible.

Since committees can have up to $1/2$ corrupt members, then it is not unlikely that all $\log(m)$ responsible nodes are corrupt. Due to this possibility, this is not a secure protocol, but in fact, none of the routing or gossiping protocols presented in the original Rapidchain paper, nor its derivation in this thesis, are secure. The only way to ensure succesful routing would be to have $mf + 1$ responsible nodes, but this would have a huge network penalty. In fact, only one responsible node needs to be honest to successfully send a transaction, given the success of the routing protocol.

We can somewhat mitigate this issue if none of the responsible nodes send the transactions by allowing C to send the cross-committee transactions again after some iterations, if no response has been received. C_{other} can then detect malicious behaviour if the responsible nodes did not send the cross committee transaction response. This approach would however be very dependent on an incentive scheme and is therefore outside the scope of this study.

A cross committee transaction is not verifiable by C_{out}

How does C_{out} verify that a completed cross committee transaction was in fact included in an accepted block in C_{in}? The authors did not discuss or present this problem.

If a committee does not verify that a completed cross-committee transaction is valid, then any malicious member of C_{in} can pose as a responsible node, as defined in definition 7, and send invalid or inconsistent transactions to C_{out}. C_{out} would have no way of knowing which return message would be correct. A simple solution would be to make all committee members send all cros-committee transactions responses, to their target committees. But this would incur a heavy performance and delay penalty, and would make all nodes in a committee potentially communicate with a large number of committees. A better solution would be to attach some kind of proof to cross-committee transaction responses that verifies that the transaction is included into the blockchain of C_{in}. If the proof is valid then a single honest responsible node is enough to ensure that C_{out} can verify the transaction.

> **Definition 8** (Cross committee transaction verifiability problem)**.** A committee C_{out} must verify that a cross committee transaction response t_i is valid. The problem is split into the two following verification problems:
>
> (1) C_{out} must verify that t_i is included in the block $b_{i_{in}}$ belonging to C_{in}.
> (2) C_{out} must verify that the block $b_{i_{in}}$ is accepted and therefor immutably added to the blockchain of committee C_{in}.

The problem defined by definition 8 is solved by the novel contribution Proof of Consensus, which is presented in the next section.

4.5 Proof of Consensus

4.5.1 Solving the cross committee transaction verificability problem

This section will address the cross-committee transaction verificability problem, as presented in definition 8.

(1) C_{out} must verify that t_i is included in the block $b_{i_{in}}$ belonging to C_{in}

To solve this problem we must create a block construction where its ID is deterministicly created by its content. If a single bit in the block's content is changed, then its ID will also change. Committees vote only on a block header that includes the block's ID. So after a block is accepted in a committee the block's content can never change.

Assuming the existence of a valid and verifiable signature set, as described in definition 11.

A simple solution would be send the entire block $b_{i_{in}}$, with a valid signature set, to C_{out}, The member of C_{out} can then easily see that transaction t_i is included in its transaction set, and if the computed ID of the block is the same as the one presented, and assuming that the signature set accepts the block header, then C_{out} can be sure that the transaction is valid. This solution would however incur a heavy network penalty since blocks are relatively large in size, and since the authors of the original paper stated that most transactions in the system are cross-committee, then these relatively large blocks will be sent to a subset of committees, severely limiting the performance increase over a system where all blocks are sent to every committee.

To minimize the required size of a block we could construct a Merkle tree, as presented by Merkle [13], on the transaction set. To prove that a transaction is part of this transaction set then it is enough to have the Merkle root and the Merkle nodes of the path towards the leaf node where the transaction is located. Given such a construction you could construct a Merkle proof t_{proof} of each cross committee transaction. This proof is enough to verify that a cross committee transaction is part of the transaction set. The Merkle tree root of the transaction set is then included in the block header. To verify a transaction t_i it is now enough to have the block header and the Merkle proof of the transaction.

This block header that includes the transaction Merkle tree root, and other relevant information, is the same for every cross committee transaction proof in the same block. Therefor it is enough to send one block header to C_{out}, and a separate Merkle proof of every t_i.

The block header can be further optimized if we allow a committee to only vote on the ID of a block. Then we can construct a small hash tree of the block header, where the root of this hash tree is the block's ID. To verify a transaction you only need three values. The node in the hash tree then contains the hashes of all information except the Merkle root of the transaction set BH_h, The Merkle root of the transaction set ts_{mr}, and the Merkle proof t_{proof}. The hash of the two first values will then produce the ID of the block $ID = H(ts_{mr}||BH_h)$.

> **Definition 9** (Block ID)**.** The ID of a block is created by the following formulas where TS_{mr} is the Merkle root of the transaction set, BH_{info} is other arbitrary information concatenated and hashed, and H is a hash function.
>
> $$BH_{info} = H(\text{info}_i||\ldots||\text{info}_n) \tag{4.2}$$
>
> $$\text{Block ID} = H(TS_{mr}||BH_{info}) \tag{4.3}$$
>
> **Remark.** A block header can be represented by just the two values TS_{mr} and BH_{info}. Assuming that the output of the hash function H is of size 32 bytes, then this representation is only 64 bytes.
>
> **Definition 10** (Proof of Inclusion)**.** A proof of any transaction t belonging to any block B in any committee C can be represented by the following set
>
> $$\text{PoI} = [BH_{info}, TS_{mr}, t_{proof}] \tag{4.4}$$
>
> Where TS_{mr} is the Merkle root of the transaction set, BH_{info} is other arbitrary information concatenated and hashed $BH_{info} = H(\text{info}_1||\ldots||\text{info}_n)$, and t_{proof} is the Merkle Proof of a transaction in the transaction set TS.

But this proof only verifies that a transaction was included in a fictional block in C_{in}, how can C_{out} verify that the block was immutably added to the blockchain of C_{in}?

(2) C_{out} must verify that the block $b_{i_{in}}$ is accepted and therefor immutably added to the blockchain of committee C_{in}.

How can committee C_{out} verify that a block $b_{i_{in}}$ produced by C_{in} is valid? This can be solved by constructing a signature set consisting of the accept messages of a block ID, the public key, and a valid signature from that public key of the accept message and block ID. C_{out} must be able to verify that the public keys of the signature set indeed do belong in that committee, and C_{out} must also ensure that the set of public keys in the signature set is larger that $mf + 1$. Only then can committee C_{out} ensure that the block from C_{in} is valid.

As was explained in Section 4.2, the reconfiguration block includes all ID's and committee memberships, and is sent to every node in the system every epoch. Committee C_{out} can therefor easily check that the set of public keys in the signature set S_{C_i} belongs to committee C_{in} in epoch i by checking the reconfiguration block B_{r_e} of the current epoch.

Definition 11 (Signature set)**.** The signature set S of an accepted block B in iteration i by committee C_{in}, denoted by S_{C_i}, is defined as a set of s_i from $i = 0, \ldots, \#$(number of signatures).

$$s_i = [\text{Tag=accept, pk=public key, signature from pk of (Block ID, Tag)}] \tag{4.5}$$

4.5.2 Proof of Consensus

In this section we will combine the results of Section 4.5.1 to solve the cross committee transaction verificability problem with the novel contribution Proof of Consensus.

Definition 12 (Proof of Consensus)**.** Any transaction t, included in any accepted block B_i in any committee C, has the following Proof of Consensus (PoC) t_{PoC}.

$$t_{PoC} = [\text{PoI}, S_{C_i}] \tag{4.6}$$

Where PoI is the Proof of Inclusion defined in definition 10 and S_{C_i} is the signature set of the accepted block B_i defined in definition 11.

Theorem 1. A valid Proof of Consensus t_{PoC} verifies that the transaction t belongs in an accepted and therefore immutable block B_i in the committee C, if and only if every committee is honest.

Proof. The Proof of Inclusion, as presented in definition 10, is correct if the hash function H is collision-proof and if the Merkle proof t_{proof} presented by Merkle [13] is correct.

The signature set S_{C_i}, as described in definition 11, can be verified by

(1) Verifying that the public keys in S_{C_i} belong in committee C in the current reference block B_{r_e}
(2) Verifying that at least $mf + 1$ signatures, that are valid against item (1), are also valid against the Block ID, as presented in the Proof of Inclusion, and the tag, as presented in the signature s_i.

If the committee is not honest then it could produce an invalid block with more than $mf + 1$ valid signatures. Since all conditions presented in this

proof are still valid in that scenario, the Proof of Consensus would be invalid.

If all the conditions are valid, then it is verified that the committee C has accepted the block B_i, that contains transaction t. The block, and subsequently the transaction, is therefor immutably added to the blockchain in C. □

Corollary 1.1. A Proof of Consensus t_{PoC} can be verified by any node in the system.

Proof. The required information to prove a Proof of Consensus is only the information inside the Proof itself and the reconfiguration block B_{r_e} with the same epoch number as the transaction t. Every node in the system stores all reconfiguration blocks B_{r_e}, and therefore is able to prove any Proof of Consensus in any epoch. □

The Proof of Consensus is of a relatively large size, since it has to be sent with every cross-committee transaction response, but it is required to verify a cross-committee transaction response.

Only the Merkle proof t_{proof} of its inclusion in the transaction set is unique to every cross committee transaction response in a single iteration. It is therefore enough to only send one complete Proof of Consensus to each committee involved. Every cross committee transaction response can then be verified with only t_{proof} and the first t_{PoC} in every iteration.

4.5.3 Applications on the cross committee transaction protocol

The Proof of Consensus construction enables the use of verifiable cross committee transactions.

There is no need to include a proof of consensus when sending derived cross-committee transactions from C_{out} because any request to do so can only come from a new and valid transaction from a user.

However, when the committees C_{in} send the cross committee transaction responses back to C_{out}, then C_{out} must verify that the transactions are valid, or else there could be potential double spending.

Therefor every cross committee transaction response must have a Proof of Consensus attached. As described in the previous subsection, we only send the first t_{PoC} to C_{out} and the rest of the cross-committee transactions responses that are to be sent to C_{out} only include the Merkle proof of inclusion in the transaction set t_{proof}.

4.5.4 Size

In this subsection, if the following assumptions are made: The output of hash function H is 32 bytes, the block header can be represented by $2 \times H$ outputs, L_{TS} is the length of the transaction set, a Merkle proof can be represented by a path of $\log_2$(length of transaction set) nodes where each leaf node is 32 bytes, a valid signature can be represented by 96 bytes, a block needs $mf + 1$ accept messages: then the minimum size of a Proof of Consensus can be calculated by the following formula:

$$\text{PoC}_{\text{minsize}} = 64\text{B} + \log_2(L_{TS} \times 32\text{B}) + 96\text{B} \times (m \times f + 1) \tag{4.7}$$

The only variable in the equation above that may differ from iteration to iteration, assuming that all committee members are online and therefore m is constant, is the length of the transaction set. Assuming that a block is always filled with the maximum amount of transactions, ignoring the size of the block header, and assuming every transaction uses one Proof of Consensus, then one could use the following formula to calculate the length of the transaction set, where t_{avgsize} is the average transaction size without a Proof of Consensus attached and B is the block size.

$$L_{TS} = \frac{B}{t_{\text{avgsize}} + \text{PoC}_{\text{minsize}}} \tag{4.8}$$

Substituting L_{TS} in equation 4.7 with the result from equation 4.8, and solving for $\text{PoC}_{\text{minsize}}$ yields the following equation:

$$\text{PoC}_{\text{minsize}} = \frac{W(2^{165+96\times f\times m+t_{\text{avgsize}}} \times B \times \log(2))) - t_{\text{avgsize}} \times \log(2)}{\log(2)} \tag{4.9}$$

Where W is the Lambert W function, also called the product log function, introduced by Lambert [15].

As discussed in Section 4.5.3, as the number of cross committee transactions grow larger, we can amortize the cost by only sending one block header and signature set per iteration to each committee that requires it. Every cross committee can then only be represented by the size of the Merkle proof.

Without block header and signature set, with the use of equation 4.8, and again solving for $\text{PoC}_{\text{minsize}}$ we get the equation:

$$\text{PoC}_{\text{minsize}} = \frac{W(2^{5+t_{\text{avgsize}}} \times B \times \log(2))) - t_{\text{avgsize}} \times \log(2)}{\log(2)} \tag{4.10}$$

Chapter 5

Results

All the data presented in this chapter comes from the measurements of a single experiment, except Figure 5.2, Table 5.4 and Table 5.5, which present the analytical results of the equation presented in section 4.5.4.

5.1 Test setup

The experiment ran on 16 c5a.4xlarge instances on the AWS cloud infrastructure [17]. Each c5a.4xlarge had 16 virtual CPUs @ up to 3.3 GHz, with 32 GiB of ram. Each instance ran 8 nodes. A coordinator node, used to initiate the experiment, emulate the bootstrap protocol, log results, and generate transactions, was placed on a separate, but similar instance. All instances was placed in the same region on the same sub network, therefor minimizing potential latency.

The original paper stated that the experiment utilized 32 Intel Xeon Phi 7210 machines with 1.3GHz processors, where each machine ran 125 nodes. These machines has 256 threads available [16], and therefor utilized approximately two threads per node. Threads are equivalent to virtual CPU's on the machines utilized in this thesis [17]. Even though two threads was utilized for each node in both implementations, the CPUs used in this implementation had the potential of reaching up to 3.3 GHz [17], almost three times as much as in the original implementation.

5.2 Implementation details

The implementation used in this thesis was written in the programming language Go[1] due to the simplicity of writing concurrent programs with heavy amounts of network usage. The source code of the implementation is open source[2].

[1] `https://golang.org/`

[2] `https://github.com/kimborgen/rapidchain`

Parameter	Value	Note
n	128	total network size
m	16	committee size (8 committees)
Total f	1/3	total adversaries
Committee f	1/2	committee adversaries
tps	10	transactions per seconds generated
κ	128	IDA Gossip shard size
Φ	80	IDA Gossip redundancy
Δ	4000	synchronous Δ
B	2^{21} bytes	block size

Table 5.1: Parameters used in the presented experiment

Mean	2.63 ms
Median	2.08 ms
Min	0.66 ms
Max	314.22 ms

Table 5.2: Measured performance when adding a Proof of Consensus

All communication is done over TCP channels, to guarantee delivery. Public and private keys, and their signatures uses normal ECDSA cryptography [26][3], with the elliptic curve secp256r1 [28]. The hash function H is the SHA-256 function, which produces an output of 256 bits = 32 bytes [27]. For more implementation details visit the documentation in the README in the open source Github repository.

The bootstrap protocol was not implemented, a coordinator node was therefor used to emulate the setup required. The reconfiguration protocol was also not implemented, and therefor there is no reference committee in the current implementation, and there is only one epoch.

The parameters used to run the experiment presented in this chapter is presented in Table 5.1.

5.3 Proof of Consensus

Proof of Consensus, which is the novel contribution presented in section 4.5, only has two potential performance impacts on the system, the computational power, measured in seconds, required to add and verify a Proof of Consensus, and the size of the Proof of Consensus. The size of the proof depends on the transaction generation method, and as discussed in chapter 6, this method has limitations in the current implementation. Due to this, an analytical approach is presented

[3]The implementation used can be found here `https://golang.org/pkg/crypto/ecdsa/`

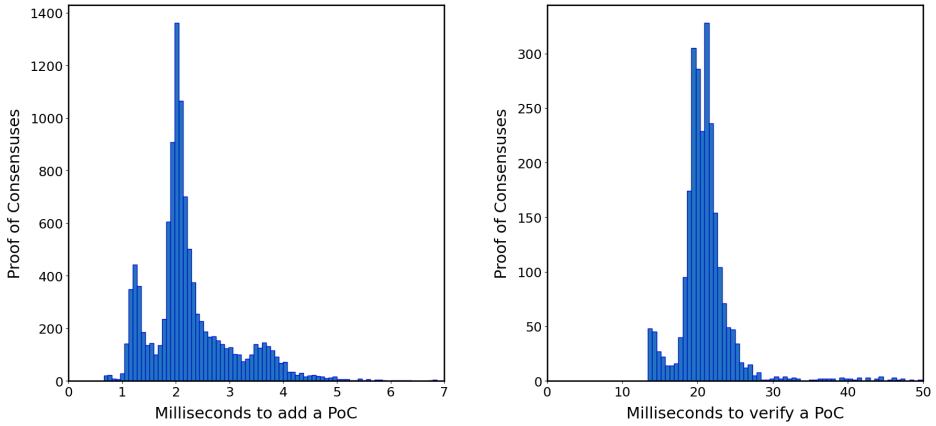


Figure 5.1: Time to add (left) and verify (right) a Proof of Consensus in a cross committee transaction response.

Mean	27.40 ms
Median	20.89 ms
Min	0.66 ms
Max	545.11 ms

Table 5.3: Measured performance when verifying a Proof of Consensus

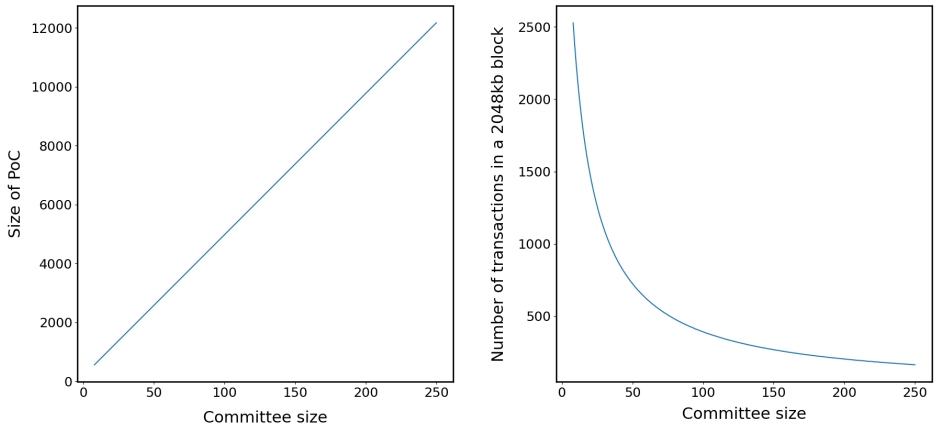


Figure 5.2: Proof of Consensus size without optimization (left). Number of transactions included in a block with a block size of 2048 kilobytes, without optimizations (right).

Committee size	Size of PoC in bytes
8	560
125	6173
250	12172

Table 5.4: Proof of Consensus size in bytes on relevant committee sizes

instead, using the equations presented in section 4.5.4.

Figure 5.1 presents the time to add and verify a Proof of Consensus, measured in milliseconds. The y-axis, where the number of Proof of Consensuses is measured, is considerably larger on the left-hand graph. This is due to the measurement method, where the $\log(m)$ response nodes, as defined in definition 7, will log the time to add a Proof of Consensus, whereas only the leader will log the time to verify the same Proof of Consensus.

Due to outliers, the x-axis on Figure 5.1 is cut off to only display relevant data. The mean, median, minimum and maximum of the same data is presented in Table 5.2 and 5.3.

The results of equation 4.9, with committee sizes $m = [8, 250]$, a block size of $B = 2048000$, an average transaction size of $t_{\text{avgsize}} = 250$ [14], and an $f = 0.5$, is presented in Figure 5.2 (left). The presented values for the size of the proof depending on committee size is then used with equation 4.8 to produce the number of transactions that can be included in a block of size 2048 kilobytes, with committee sizes $m = [8, 250]$, and is presented in Figure 5.2 (right).

Committee size	Transactions included in a block
8	2527
125	318
250	164

Table 5.5: Number of transactions included in a block of size 2048 kilobytes on relevant committee sizes

There are three relevant committee sizes. $m = 8$ is used in the experiment presented here, $m = 125$ is the one presented by Zamani *et al.* [1] when the network size is $n = 500$ (the lowest value presented in their study), and $m = 250$, which is used with the targeted network size of $n = 4000$ in the same study. The values for these committee sizes is presented in Table 5.4, for Figure 5.2 (left), and Table 5.5, for Figure 5.2 (right).

The values presented in Figure 5.2, and subsequently in Table 5.4 and 5.5, do not use the optimization presented in section 4.5.4. When the optimization method is utilized, equation 4.10 presents the estimated size of the proof. With the same values presented above, a block size of $B = 2048000$, and an average transaction size of $t_{\text{avgsize}} = 250$, equation 4.10 yields a size of $\lceil 17.9 \rceil = 18$ bytes. The amount of transactions that can be included in a block, with the same values presented above, is given by equation 4.8 and yields $\lfloor 7641.79 \rfloor = 7641$ transactions in a block.

5.4 Cross committee transactions

Figure 5.3 presents how many cross committee transactions were necessary to finish an original transaction. In the experiment presented here, a total of 1811 transactions were included in a finished block. Only 10% of those transactions could be completed without a cross committee transaction, whereas 90% required one or more cross committee transactions.

Zamani *et al.* [1] did not present the number of cross committee transactions that were involved in a transaction. To this extent, the results presented here can therefore be classified as novel although no changes in the implementation in this thesis from the original implementation by the authors should produce any changes to the presented results.

5.5 User-perceived transaction latency

Figure 5.4 presents the distribution of user-perceived transaction latency for transactions with differing amounts of cross-committee transactions. The two transaction outliers with three cross-committee transactions, as can be seen in Figure 5.3,

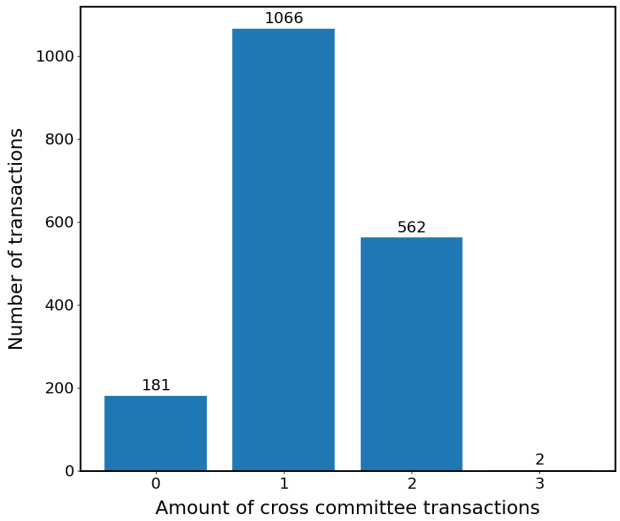


Figure 5.3: Amount of cross committee transactions involved in a transaction

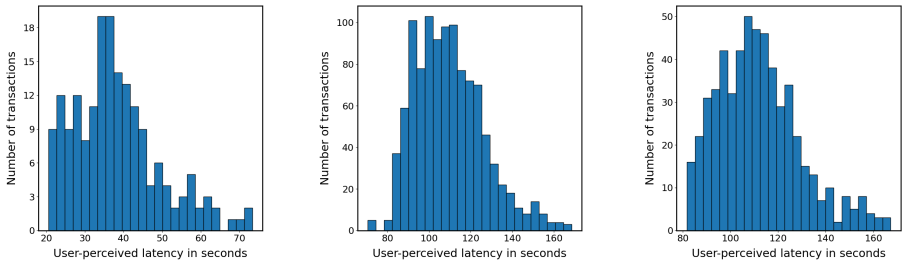


Figure 5.4: User-perceived transaction latency for transactions with no cross committee transactions (left), one cross committee transaction (middle), and two cross committee transactions (right).

Number of cross committee transactions	Measurement	Seconds
0	Mean	38.03 s
0	Median	36.34 s
1	Mean	109.69 s
1	Median	108.02 s
2	Mean	111.86 s
2	Median	110.08 s
Total	Mean	103.19 s
Total	Median	101.49 s

Table 5.6: Measured transaction latency with varying cross committee transactions.

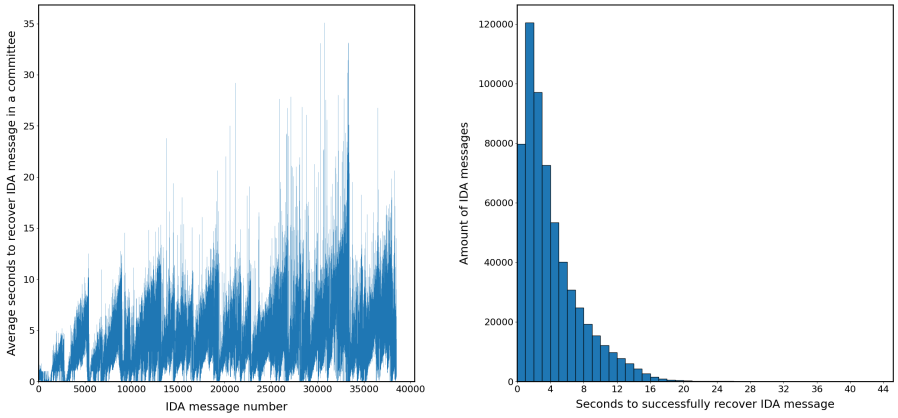


Figure 5.5: Average seconds for an entire committee to successfully recover an IDA message over time (left). Distribution of how many seconds it took to successfully recover an IDA message for a node (right).

were omitted due to non relevant results. The total mean and median was also included for comparison with the original study.

The mean and median for the graphs in Figure 5.4 are presented in Table 5.6, along with the total calculated mean and median latency for every transaction, which were of 103.19 ms and 101.49 ms respectively.

Assuming that the differing implementation details presented in this thesis do not greatly affect transaction latency, then the result presented in this section would be the best metric to compare with the original implementation presented by Zamani *et al.* [1].

Mean per node	3.63 s
Median per node	3.00 s

Table 5.7: Measured mean and median latency for completing the IDA gossip protocol for one message

5.6 IDA Gossip

Figure 5.5 presents the latency of the Information Dispersal Algorithm starting when a node initiates the IDA Gossip protocol, and ending when each node successfully reconstructs the message. Figure 5.5 (left) measures the average number of seconds it took for an entire committee to successfully reconstruct an IDA message, over time. Figure 5.5 (right) measures at each node, how many seconds it took for the node to reconstruct an IDA message, and presents the calculated distribution.

The mean and median measured latency for completing the IDA gossip protocol for one message are 3.63 s and 3.00 s respectively. This is also presented in Table 5.7

The implementation of IDA presented in this thesis differs only from the original study [1] in the choice of neighbours. The authors did not however present any performance results on the IDA protocol, so we cannot compare results directly, although if the results were to be compared they should not have been very different because the implementation difference should not affect the performance results significantly.

Since no results were presented in the original paper, the results in this section can therefor be classified as novel.

Chapter 6

Discussion

6.1 Effect of oversubscribing CPU on results

The test setup used to run the experiment presented in Chapter 5 may have imposed certain limitations on the presented results. As presented in Section 5.1, the experiment used only two virtual CPUs, the equivalent of one CPU core, for each node. Each node does however perform multiple actions simultaneously. We can see the effect of this clearly in the presented results, where the maximum values of Table 5.2, Table 5.3, and Figure 5.5 are orders of magnitude larger than the mean and median. These spikes in the data can then be explained with the scenario where the responsible CPU core chooses to prioritize other function calls in the middle of the measurement of these results.

If the system is deployed to real-world nodes, that have the capacity to use several CPU cores, then the outliers that cause spikes in the data would most likely be non existent. For this reason, the median provides a better estimate of performance than the mean, and will be used for the rest of this chapter.

6.2 Proof of Consensus

Figure 5.2, and subsequently Table 5.4 and 5.5, indicate the linear growth of a Proof of Consensus depending on committee size when not utilizing the optimization presented in Section 4.5.4. This linear growth has the effect of exponentially decreasing the number of transactions that can fit in a 2048 kilobyte block, as can be seen in Figure 5.2 (right). In the target network size, $n = 4000$, of the original study by Zamani *et al.* [1], only 164 transactions can fit in a block using this method. This presents a significant bottleneck. Therefore, the optimization presented in Section 4.5.4 must be utilized. Using optimization, only a single full proof must be sent with each block. Each subsequent proof only requires 18 bytes, and 7641 transactions can be included in a block, as presented in Section 5.3.

If every transaction in the block of 7641 transactions has 1 cross committee, and the size of a proof is 18 bytes, then the total size of every Proof of Consensus in a block would be 137.5 kilobytes. An acceptable size with respect to the full block size of 2048 kilobytes.

As we can see in Table 5.2 and 5.3, the median time to add a Proof of Consensus is 2.08 ms, whereas the median time to verify a Proof of Consensus is 20.86 ms. This difference in one order of magnitude is expected due to fact that adding a proof requires little computation, whereas when verifying a proof a node has to verify all signatures, as well as the Merkle proof, resulting in, among other non relevant computations, several invocations of the hash function H. The result presented in these tables might be questioned due to the limits of the testing setup described above. The minimum time to both add and verify presented in the same tables, indicates that the time to both add and verify a proof of consensus would be significantly lower if enough CPU resources were available. Indeed by multiplying the median times by the maximum number of transactions in a block the median time to add would be 15.9 seconds, and the median time to verify would be 159.4 seconds, for each block.

However, since the data presented here, except the median times to add and verify a proof, is based on an analytical approach, and not an experimental one, real world results may vary.

The equations in Section 4.5.2 assumed that every transaction had exactly one proof of consensus attached. This would not be the case in a real world scenario, but as discussed in the next section, due to an imperfect transaction generation method, a better real-world approximation could not be presented.

6.3 Cross committee transactions

As discussed in Section 5.4, 90% of the transactions in the presented experiment required the use of one or more cross committee transactions. This resonates somewhat with the results presented by Zamani *et al.* [1], who presented a 96.3% probability of a transaction requiring a cross committee transaction with a network size of 500 nodes, and 3 committees. But since the experiment presented in this thesis uses significantly less nodes and more committees, the results do not match perfectly.

However, in Figure 5.3 we can see that a transaction only has one or two cross committee transactions, but the experiment presented in this thesis had 8 committees. Since transaction IDs would be divided over more committees, one could also assume that a transaction would potentially have more than the observed one or two involved cross committee transactions. This may be a limitation of the transaction generation method. The transaction generation method used in the presented implementation does not come from a real world set of Bitcoin transactions, and is generated randomly. The transaction generation method may therefor have only created transactions with low amount of inputs. Due to the late

Number of cross committee transactions	Measurement	Seconds
0	Median	5.40 s
1	Median	16.20 s
2	Median	16.53 s
Total	Median	15.24 s

Table 6.1: Measured transaction latency divided by the factor 6.66.

discovery of this issue, it was not investigated. To get a better estimation, it would also be preferable to use a real world dataset.

6.4 User-perceived transaction latency

This thesis does not evaluate or measure confirmation latency. This is due to the fact that confirmation latency is fully dependant on the synchronous round time Δ. A new iteration will start as soon as the previous one is done. A new transaction received by a committee in an iteration will be included in the next one, assuming that every block will empty the transaction pool. The experiment presented in this thesis could not produce enough transactions to fill the 2048 kilobyte block. Therefore, since a transaction will always be included in the next iteration in this experiment, measuring confirmation latency is as simple as calculating the time of an iteration plus the estimated time to wait before the next iteration starts which is half the iteration time on average.

From Table 5.6 we can see that the perceived latencys of transactions containing one or two cross committee transactions are quite similar, with only about 2 ms difference. This is within expectations since the cross committee transactions are sent at the same time to the two or more committees. The only differences would be routing delay, if for example one committee is located within the routing table of the sending committee while the other committee must be routed with the inter-committee routing protocol, as well as differences in start times of iterations, because iterations are not necessarily synchronous across committees.

The experiment presented in Chapter 5 used a high Δ of 4000, compared to a Δ of 600 used by Zamani *et al.* [1]. Since the confirmation latency is fully dependant on Δ, the fact that that results are drastically worse is expected. Since this thesis increased the Δ by 6.66 times, we can calculate the expected results if we lowered the Δ to 600. Table **??** divides the results of Table 5.4 by 6.66 for a better comparison.

Zamani *et al.* [1] did not present results with a lower network size than 500. The network size of 128 presented in this experiment can therefore not be fully compared to the original. However one can expect that lower network sizes produce lower transaction latency. The lowest latency presented by Zamani *et al.* [1] was, with a network size of 500 with 3 committees, 8.04 seconds confirmation latency

and 32.2 seconds user-perceived latency, where these two values only increased with higher network sizes. The original study did not present results that differ between transactions of differing number of cross committee transactions, therefore it is only possible to compare the total median delay and not the individual delays presented in this thesis. The result of 15.24 ms, with a network size of 128, against the original result of 32.2 seconds, with a network size of 500, is not outside the realm of possibility. The user perceived latency presented by the Zamani *et al.* [1], was only stable after a network size of 1000, therefore it is impossible to calculate a formula for this latency for a better comparison.

Since the experiment did not allow any other measurement to be directly compared with the results presented by Zamani *et al.* [1], the comparison of user-perceived latency is the best comparison this thesis can make. As was discussed above, it is impossible to compare the latency directly, but the perceived latency was well within the realm of possibility. Therefor we can conclude that the implementation presented in this thesis is most likely similar to the one presented by Zamani *et al.* [1] although this cannot be proven.

6.5 IDA Gossip

The size of committees was only 16 nodes. The lowest committee size presented by Zamani *et al.* [1] is 125. Due to this difference, any results presented could not be true in a larger committee. This section will however look at trends that may be applied to larger committee sizes.

> **Remark.** The implementation presented in this thesis starts each iteration in every committee at almost identical times. Due to synchronous rounds, the rounds should therefore also be synchronized across committees at the start. Small changes in delay may have consequences, and rounds across committees should therefore not be synchronized for long. Rounds are not synchronized in either this implementation or the original one. The synchronous rounds across committees in the start may therefore produce patterns in the data that may not be true after a while.

As one can see in Figure 5.5 (left), the average time to complete the IDA Gossip protocol for a committee increases, and then suddenly drops. This pattern continues to be observed, but becomes more unclear the more IDA messages are sent. The only component outside the protocol that could invoke the use of the IDA protocol is the transaction generator, but the transaction generator was designed to send a constant amount of transactions per second, randomly to any node in the system, and should therefore not produce such a pattern. Since iterations across committees are synchronized in the start, the increase and rapid decrease should therefor encapsulate the natural iterations in committees.

The above-mentioned pattern is more clear in the start, since no cross committee transactions exist in the system yet, and the only components invoking the IDA

protocol at that time would be transactions and the proposing of blocks. After a few iterations one can still see the pattern, but it may be shifted due to iterations across committees starting to become unsynchronized.

The spiking patterns at the end of iterations may be due to cross committee transactions being routed to their target committees. Each receiving committee of such transactions receives a minimum of $\log(m)$ messages of the same transaction from the responsible nodes in the sending committee, and possibly more if the routing protocol is utilized. Each of those messages would invoke the IDA protocol, even though they are equal. This is required for security, but if t different cross committee transactions are sent to the same targeting committee from just one sending committee, then a minimum of $t \times \log(m)$ messages need the invoking of the IDA Gossip protocol, from only one sending committee (excluding invoking the IDA protocol from normal transactions). If every committee sends an average of t cross committee transactions to each other committee in every iteration, then the minimum of uses of the IDA protocol per committee would be $n/m \times t \times \log(m)$. This may be the reason for large spikes of up to 34 second delays in the IDA protocol.

The issue presented immediately above should be solved by batching cross committee transactions, as Zamani *et al.* [1] presented. If these transactions are batched together, then the IDA protocol should only be invoked $n/m \times \log(m)$ per iteration, and is therefore not susceptible to the amount of cross-committee transactions, but rather only to the size of the network and committee sizes. The late discovery of this issue resulted in batching not being implemented.

The analysis of the results discussed in this section showed the existence of a bottleneck caused by the impact of the cross committee transaction protocol on the IDA protocol. The issue is however easily solvable with batching. Batching is not a novel contribution, but the argument for why batching is required becomes more transparent with these results.

6.6 limitations

One large limitation of this study is the lack of computational resources required for a proper comparison with the original study by Zamani *et al.* [1]. In testing, it was impossible to launch more nodes per CPU than the ones presented in the experiment in this thesis because the system failed due to the halting of too many nodes. Since the maximum number of nodes the testing setup presented here could run was 128, and the results with the lowest amounts of nodes in the original study used 500 nodes, it is impossible to compare fully the two implementations. The source code of the implementation presented by Zamani *et al.* [1] is not available and can therefore not be used for comparison. Nonetheless, the results discussed earlier in this chapter indicate similar trends in the data which can be compared.

Another large limitation was the runtime of the experiment. The experiment presen-

ted in this thesis inexplicably ended after about 5 minutes. This may have been due to the spiking pattern observed, causing huge bottlenecks that would halt nodes and therefor break the synchronicity assumption. Due to limitations in time, and the late discovery of possible bottlenecks, the issue could not be solved. The previous sections in this chapter did however present possible solutions to these bottlenecks, and once implemented, and run on a testing setup where multiple virtual CPU's can be assigned to each node, I believe that the experiment could be run for the expected time.

Due to the limitations in runtime, as presented in the previous paragraph, measurement of more relevant metrics could not be performed. The most interesting metric in the original study was, in my opinion, the scalability factor. The transaction generation method was capped at producing at most 10 transactions per seconds, otherwise the bottlenecks presented in the previous sections would be large enough to only allow a successfully runtime. Due to the 10 transactions per second limit, an honest comparison of the scalability factor could not be presented.

Due to the limitations presented in this sections, the artificial networking delay of 100ms as used in the original study was not implemented. Malicious behaviour was also not implemented. The reason to not implement these importunity details was mainly done to better observe the effect of the bottlenecks directly, to discover the cause of the limitations, but due to time limitations, the solution to these problem could not be located in time.

The Δ was also set 6.66 times larger than the original Δ due to the bottlenecks presented in this chapter. Lowering the Δ resulted in shorter runtimes with less completed transactions. In order to gather interesting results to present in this thesis the Δ had to be of such a large size. This further indicates that the synchronicity assumption did not hold in the implementation presented in this thesis.

The lack of the source code of the original implementation by Zamani *et al.* [1], and the lack of implementation details presented in the original study, imposes a significant limitation on the comparability between the two implementations. Several details had to be clarified, or derived, in Chapter 4, and may differ from the original implementation.

The scope of this study was also limited to only a subset of the full original implementation. Several components, such as the bootstrap protocol, reconfiguration protocol and the responsibilities of the reference committee were not investigated. Sparsification of the Merkle trees in the IDA Gossip protocols was also not implemented. Therefor the implementation and results presented in this thesis cannot be classified as a full replicability study. These protocols were discussed in detail in the original paper by Zamani *et al.* [1], whereas the components implemented in this thesis, especially the cross-committee transaction protocol, were not equally rigorously discussed compared with those in the original study, and the results and discussion presented here, except the novel contribution Proof of Consensus, can therefore be classified as an extension, or complement, of the original

study.

A significant limitation on the literature review presented in Chapter 3, was the choice of search method. The assumption that every relevant study on committee based sharding would reference Rapidchain is very strong. The inherent limitation in the choice of search query is that only studies published after the original Rapidchain paper was published would be found, although committee based sharding existed well before the original Rapidchain paper [23].

As was presented in the literature review in Chapter 3, the work of Homoliak *et al.* [63] should be analysed against any implementation of the Rapidchain protocol, but the implementation presented in this thesis was not. This was mainly due to an incomplete implementation, and time limitations.

Chapter 7

Conclusion

7.1 Future work

The most important future work would be to solve the bottlenecks presented in the previous chapter. Multiple possible solutions are presented, and these form a primer for future work. Another obvious future work would be to expand the implementation presented in this thesis to include the missing components, such that a full replication study can be performed.

Section 6.2 presents and discusses the potential problem of the time to add and verify Proof of Consensuses in a full block of 7641 transactions. The verification time of 159 seconds should be decreased by at least two orders of magnitude so as to avoid a bottleneck in the system. The implementation of the Proof of Consensus verification method is sequential, but each Proof of Consensus can be verified in parallel. Other optimizations in code can also be utilized.

Another important future work would be to analyze the size of a Proof of Consensus with different amounts of cross committee transactions involved. The analytical results in this thesis assume that every transaction has exactly one Proof of Consensus, but as was shown in Section 6.3, the number of cross committee transactions, and therefor Proof of Consensuses, differ. Due to an incomplete transaction generation method, the true amount of cross-committee transactions is assumed to differ from the presented results in Chapter 5. An important future work is therefore to analyze how many cross committee transactions would be required, on average, for each transaction in a real world dataset. The results from this could then be used to create a better analytical approach than the one presented in Section 4.5.4. These results should be complemented with an experimental measurement as well.

An important metric in the original study that was not discussed in this thesis, but that should be investigated further, would be the impact of nodes joining or leaving committees in an epoch. The original study authors only tested the performance impact of up to 10 nodes joining and leaving the entire system, with

a network size of $n = 4000$. The assumption that only 10 nodes will join or leave the system in an epoch time of 24 hours may not be practical for a real world scenario. The performance impact also seems to be exponential, but the authors did not present enough data to reach such a conclusion. An important future work is therefore to implement the reconfiguration protocol, and test the impact of more nodes joining and leaving the system during an epoch.

As presented in Figure 5.2, the size of the the Proof of Consensus linearly grows with committee size. The only part of the equation 4.7 that grows with respect to the committee size m, is the size of the signature set. A future work could then be to decrease the size of this signature set, but since it was proposed to amortize the full size of the Proof of Consensus by only sending the full Proof of Consensus once, the relevant performance impact of decreasing the signature set may not be large. However, to give a primer for future research, one could employ a signature aggregation scheme, such as Schnorr [18] or BLS signatures [19], to shorten the size of the signature set to only one verifiable signature.

Another important question that was not answered in this thesis would be if the synchronicity assumption is practical for real-world use. The synchronicity assumption states that a message is delivered to all nodes within a know time Δ. This cannot be true in a real world scenario. An important future work would therefore be to change the consensus protocol to allow for semi-synchronous rounds, where a committee can recover from a potential spike in delay. Optimistic timeouts can also be employed. A protocol to dynamically change Δ either in epochs, or in iterations, is also required.

The potential of a Denial-of-Service attack on the routing protocol, as discussed in Section 4.2, should also be investigated given higher computational resources.

Three important questions that were uncovered in the literature review were not addressed in this thesis. The potentiality of bias in the distributed random generation method, the construction of an incentive scheme, and the analysis of the Rapidchain implementation in the work Homoliak *et al.* [63]. These issues are therefore left to future work.

7.2 Conclusion

This thesis presented identified limitations and issues on the state of the art solution Rapidchain, as presented by Zamani *et al.* [1]. These limitations and issues are discussed in Chapter 4, and possible solutions or clarifications were presented. The main limitations of the Rapidchain protocol were found to be its cross-committee transaction protocol, where no cross-committee transaction could be verified across committees, causing the potential of double-spending. A solution to this cross- committee transaction verificability problem was presented in Section 4.5. The solution creates a new construction called Proof of Consensus, which enables any node in the Rapidchain system to verify that any transaction is included in any finished block in any committee. The verificability of a Proof of Consensus

was proved in Thereom 1, and subsequently by Corollary 1.1. In Chapter 5 and 6 it was shown that the size of these proofs could be as low as 18 bytes. Under certain assumptions this would indicate that only 137.5 kilobytes of proofs would need to be added to a block with size 2048kb. However, the time required to verify a large amount of these proofs could be a potential bottleneck. Several possible solutions were discussed, but the implementation of such solutions was left to future work. The rest of the results and discussion presented in the same chapters indicated that the results from the experiment presented in this thesis were not enough to represent a full comparison, but the results did nevertheless indicate similar trends.

7.3 Acknowledgments

I would like to thank Mariusz Nowostawski, my supervisor, for his invaluable guidance in both my specialization project [2], and this master thesis.

I would also like express my gratitude towards my step father for helping me with grammatical corrections, and the rest of the family for providing support.

Finally I would like to thank my good friends at the university for providing good humour[1] and companionship in the unprecedented global events that occurred during the semester when this masters thesis was written.

[1] aaaaa

Bibliography

[1] M. Zamani, M. Movahedi and M. Raykova, *Rapidchain: Scaling blockchain via full sharding*, Cryptology ePrint Archive, Report 2018/460, `https://eprint.iacr.org/2018/460`, 2018.

[2] K. A. T. Borgen, *A structured literature review on the topic of scalability and performance of public blockchains*, 2019.

[3] S. Nakamoto, 'Bitcoin: A peer-to-peer electronic cash system', Manubot, Tech. Rep., 2019.

[4] G. Wood *et al.*, 'Ethereum: A secure decentralised generalised transaction ledger', *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[5] W. Hoeffding, 'Probability inequalities for sums of bounded random variables', in *The Collected Works of Wassily Hoeffding*, Springer, 1994, pp. 409–426.

[6] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer and B. Ford, 'Scalable bias-resistant distributed randomness', in *2017 IEEE Symposium on Security and Privacy (SP)*, Ieee, 2017, pp. 444–460.

[7] D. Boneh, J. Bonneau, B. Bünz and B. Fisch, 'Verifiable delay functions', in *Annual international cryptology conference*, Springer, 2018, pp. 757–788.

[8] P. Dworzanski. (). A note on committee random number generation, commit-reveal, and last-revealer attacks., [Online]. Available: `http://paul.oemm.org/commit_reveal_subcommittees.pdf` (visited on 15/05/2020).

[9] L. Ren, K. Nayak, I. Abraham and S. Devadas, 'Practical synchronous byzantine consensus', *arXiv preprint arXiv:1704.02397*, 2017.

[10] 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*, Springer, 2002, pp. 53–65.

[11] R. W. Hamming, 'Error detecting and error correcting codes', *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[12] P. Mahlmann and C. Schindelhauer, 'Distributed random digraph transformations for peer-to-peer networks', in *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, 2006, pp. 308–317.

[13] R. C. Merkle, 'A digital signature based on a conventional encryption function', in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378, ISBN: 978-3-540-48184-3.

[14] J. Göbel and A. E. Krzesinski, 'Increased block size and bitcoin blockchain dynamics', in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, 2017, pp. 1–6.

[15] J. H. Lambert, 'Observationes variae in mathesin puram', *Acta Helvetica*, vol. 3, no. 1, pp. 128–168, 1758.

[16] *Xeon phi*, Apr. 2020. [Online]. Available: `https://en.wikipedia.org/wiki/Xeon_Phi`.

[17] *Amazon ec2 instance types - amazon web services*, `https://aws.amazon.com/ec2/instance-types/#instance-details`, (Accessed on 07/14/2020).

[18] G. Maxwell, A. Poelstra, Y. Seurin and P. Wuille, 'Simple schnorr multi-signatures with applications to bitcoin', *Designs, Codes and Cryptography*, vol. 87, no. 9, pp. 2139–2164, 2019.

[19] D. Boneh, M. Drijvers and G. Neven, 'Bls multi-signatures with public-key aggregation',

[20] T. zilliqa foundation, *Zilliqa*, `https://www.zilliqa.com/`, (Accessed on 07/15/2020).

[21] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta and B. Ford, 'Omniledger: A secure, scale-out, decentralized ledger via sharding', in *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 583–598.

[22] B. Koteska, E. Karafiloski and A. Mishev, 'Blockchain implementation quality challenges: A literature', in *SQAMIA 2017: 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, 2017, pp. 11–13.

[23] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert and P. Saxena, 'A secure sharding protocol for open blockchains', in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30.

[24] *Google scholar*, `https://scholar.google.com/`, (Accessed on 07/15/2020).

[25] S. Keele *et al.*, 'Guidelines for performing systematic literature reviews in software engineering', Technical report, Ver. 2.3 EBSE Technical Report. EBSE, Tech. Rep., 2007.

[26] D. Johnson, A. Menezes and S. Vanstone, 'The elliptic curve digital signature algorithm (ecdsa)', *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[27] *Sha-2 - wikipedia*, `https://en.wikipedia.org/wiki/SHA-2`, (Accessed on 07/15/2020).

[28] S. Turner, D. Brown, K. Yiu, R. Housley and T. Polk, 'Elliptic curve cryptography subject public key information', *RFC 5480 (Proposed Standard)*, 2009.

[29] S. King and S. Nadal, 'Ppcoin: Peer-to-peer crypto-currency with proof-of-stake', *self-published paper, August*, vol. 19, p. 1, 2012.

[30] L. Lamport, R. Shostak and M. Pease, 'The byzantine generals problem', in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 203–226.

[31] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi and J. Stern, 'Addendum to "scalable secure storage when half the system is faulty"[inform. comput. 174 (2)(2002) 203–213]', *Information and Computation*, vol. 205, no. 7, pp. 1114–1116, 2007.

[32] I. S. Reed and G. Solomon, 'Polynomial codes over certain finite fields', *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[33] E. Kokoris-kogias, 'Robust and Scalable Consensus for Sharded Distributed Ledgers',

[34] M. Wang and Q. Wu, 'Lever: Breaking the Shackles of Scalable On-chain Validation',

[35] M. El-hindi, 'BlockchainDB - A Shared Database on Blockchains', vol. 12, no. 11, pp. 1597–1609,

[36] Harmony, 'Harmony', *Journal of the American Medical Association*, vol. XXXIII, no. 1, pp. 45–46, 1899, ISSN: 23768118. DOI: 10.1001/jama.1899.02450530051008.

[37] J. Suzuki and T. Suda, 'Design and Implementation of an Scalable Infrastructure for Autonomous Adaptive Agents', *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, vol. 15, no. 2, pp. 594–603, 2003.

[38] J. Wilsdon, 'The politics of small things: Nanotechnology, risk, and uncertainty', *IEEE Technology and Society Magazine*, vol. 23, no. 4, pp. 16–21, 2004, ISSN: 02780097. DOI: 10.1109/MTAS.2004.1371634. arXiv: arXiv:1011.1669v3. [Online]. Available: http://waset.org/publications/14223/soil-resistivity-data-computations-single-and-two-layer-soil-resistivity-structure-and-its-implication-on-earthing-design%7B%5C%%7D0Ahttp://www.jo-mo.com/fadoohelp/data/DotNet/Ethical%20securty.pdf%7B%5C%%7D0Ahttp://link.springer.com/10.10.

[39] M. H. Manshaei, M. Jadliwala, A. Maiti and M. Fooladgar, 'A Game-Theoretic Analysis of Shard-Based Permissionless Blockchains', *IEEE Access*, vol. 6, pp. 78 100–78 112, 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2884764. arXiv: 1809.07307.

[40] M. Ahmed and K. Kostiainen, 'Don't Mine, Wait in Line: Fair and Efficient Blockchain Consensus with Robust Round Robin', 2018. arXiv: 1804.07391. [Online]. Available: http://arxiv.org/abs/1804.07391.

[41] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan and P. Viswanath, 'PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously', pp. 1–12, 2018. arXiv: 1809.10361. [Online]. Available: http://arxiv.org/abs/1809.10361.

[42] S. Das, A. Kolluri, P. Saxena and H. Yu, *(Invited Paper) on the Security of Blockchain Consensus Protocols Sourav*. Springer International Publishing, 2018, vol. 1, pp. 146–167, ISBN: 9783030051716. DOI: 10.1007/978-3-030-05171-6. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-05171-6%7B%5C_%7D8.

[43] Z. Fidelman, 'A Generic Sharding Scheme for Blockchain Protocols', no. June, 2019.

[44] J. Wang, Y. Zhou, X. Li, T. Xu and T. Qiu, 'A Node rating based sharding scheme for blockchain', *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2019-December, pp. 302–309, 2019, ISSN: 15219097. DOI: 10.1109/ICPADS47876.2019.00050.

[45] H. Liu, F. Shen, Z. Liu, Y. Long, Z. Liu, S. Sun, S. Tang and D. Gu, 'A secure and practical blockchain scheme for IoT', *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, pp. 538–545, 2019. DOI: 10.1109/TrustCom/BigDataSE.2019.00078.

[46] I. Homoliak, S. Venugopalan, Q. Hum and P. Szalachowski, 'A security reference architecture for blockchains', *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*, pp. 390–397, 2019. DOI: 10.1109/Blockchain.2019.00060. arXiv: 1904.06898.

[47] E. Kokoris-kogias, A. Spiegelman, D. Malkhi and I. Abraham, 'Bootstrapping Consensus Without Trusted Setup : Fully Asynchronous Distributed Key Generation', no. V, pp. 1–22, 2019. [Online]. Available: https://eprint.iacr.org/2019/1015.pdf.

[48] T. Chitra and U. Chitra, 'Committee Selection is More Similar Than You Think: Evidence from Avalanche and Stellar', 2019. arXiv: 1904.09839. [Online]. Available: http://arxiv.org/abs/1904.09839.

[49] G. Avarikioti, E. Kokoris-Kogias and R. Wattenhofer, 'Divide and Scale: Formalization of Distributed Ledger Sharding Protocols', pp. 15–17, 2019. arXiv: 1910.10434. [Online]. Available: http://arxiv.org/abs/1910.10434.

[50] B. Bünz, L. Kiffer, L. Luu and M. Zamani, 'FlyClient: Super-Light Clients for Cryptocurrencies', *20'S&P*, pp. 1–31, 2019. [Online]. Available: https://eprint.iacr.org/2019/226.pdf.

[51] J. Wang and H. Wang, 'Monoxide: Scale out blockchain with asynchronous consensus zones', *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019*, pp. 95–112, 2019.

[52] A. Hafid, A. S. Hafid and M. Samih, 'New mathematical model to analyze security of sharding-based blockchain protocols', *IEEE Access*, vol. 7, pp. 185 447–185 457, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2961065.

[53] A. Manuskin, M. Mirkin and I. Eyal, 'Ostraka: Secure Blockchain Scaling by Node Sharding', 2019. arXiv: 1907.03331. [Online]. Available: http://arxiv.org/abs/1907.03331.

[54] A. Bartolomey, 'Progress on the Use of Sharding to Enhance Blockchain Scalability', pp. 1–15, 2019.

[55] A. Sonnino, S. Bano, M. Al-Bassam and G. Danezis, 'Replay Attacks and Defenses Against Cross-shard Consensus in Sharded Distributed Ledgers', 2019. arXiv: 1901.11218. [Online]. Available: http://arxiv.org/abs/1901.11218.

[56] B. Choi, J. Y. Sohn, D. J. Han and J. Moon, 'Scalable Network-Coded PBFT Consensus Algorithm', *IEEE International Symposium on Information Theory - Proceedings*, vol. 2019-July, pp. 857–861, 2019, ISSN: 21578095. DOI: 10.1109/ISIT.2019.8849573.

[57] S. Gupta, J. Hellings and M. Sadoghi, 'Scaling Blockchain Databases through Parallel Resilient Consensus Paradigm', no. 2, 2019. arXiv: 1911.00837. [Online]. Available: http://arxiv.org/abs/1911.00837.

[58] S. Kadhe, J. Chung and K. Ramchandran, 'SeF: A Secure Fountain Architecture for Slashing Storage Costs in Blockchains', 2019. arXiv: 1906.12140. [Online]. Available: http://arxiv.org/abs/1906.12140.

[59] M. J. Amiri, D. Agrawal and A. E. Abbadi, 'SharPer: Sharding Permissioned Blockchains Over Network Clusters', pp. 1–25, 2019. arXiv: 1910.00765. [Online]. Available: http://arxiv.org/abs/1910.00765.

[60] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias and W. J. Knottenbelt, 'SoK: Communication Across Distributed Ledgers', pp. 1–23, 2019.

[61] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn and G. Danezis, 'Sok: Consensus in the age of blockchains', *AFT 2019 - Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, no. Section 4, pp. 183–198, 2019. DOI: 10.1145/3318041.3355458.

[62] G. Wang, Z. J. Shi, M. Nixon and S. Han, 'Sok: Sharding on blockchain', *AFT 2019 - Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp. 41–61, 2019. DOI: 10.1145/3318041.3355457.

[63] I. Homoliak, S. Venugopalan, Q. Hum, D. Reijsbergen, R. Schumi and P. Szalachowski, 'The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses', pp. 1–44, 2019. arXiv: 1910.09775. [Online]. Available: http://arxiv.org/abs/1910.09775.

[64] H. Dang, T. T. A. Dinh, D. Loghin, E. C. Chang, Q. Lin and B. C. Ooi, 'Towards scaling blockchain systems via sharding', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 123–140, 2019, ISSN: 07308078. DOI: 10.1145/3299869.3319889. arXiv: 1804.00399.

[65] N. Chawla, H. W. Behrens, D. Tapp, D. Boscovic and K. S. Candan, 'Velocity: Scalability Improvements in Block Propagation Through Rateless Erasure Coding', *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, pp. 447–454, 2019. DOI: 10.1109/BLOC.2019.8751427.

[66] S. Kim, J. Song, S. Woo, Y. Kim and S. Park, *Gas consumption-aware dynamic load balancing in ethereum sharding environments*, 2019. DOI: 10.1109/FAS-W.2019.00052.

[67] Y. Liu, J. Liu, Z. Zhang and H. Yu, 'A fair selection protocol for committee-based permissionless blockchains', *Computers and Security*, vol. 91, 2020, ISSN: 01674048. DOI: 10.1016/j.cose.2020.101718.

[68] Y. Xu, Y. Huang, J. Shao and G. Theodorakopoulos, 'A flexible n/2 adversary node resistant and halting recoverable blockchain sharding protocol', *Concurrency Computation*, no. December 2019, pp. 1–13, 2020, ISSN: 15320634. DOI: 10.1002/cpe.5773.

[69] A. Hafid, A. S. Hafid and M. Samih, 'A methodology for a probabilistic security analysis of sharding-based blockchain protocols', *Advances in Intelligent Systems and Computing*, vol. 1010, pp. 101–109, 2020, ISSN: 21945365. DOI: 10.1007/978-3-030-23813-1_13.

[70] Y. Xu and Y. Huang, 'An n/2 byzantine node tolerate blockchain sharding approach', *Proceedings of the ACM Symposium on Applied Computing*, pp. 349–352, 2020. DOI: 10.1145/3341105.3374069. arXiv: 2001.05240.

[71] M. Zhang, J. Li, Z. Chen, H. Chen and X. Deng, 'CycLedger: A Scalable and Secure Parallel Protocol for Distributed Ledger via Sharding', 2020. arXiv: 2001.06778. [Online]. Available: http://arxiv.org/abs/2001.06778.

[72] S. Woo, J. Song, S. Kim, Y. Kim and S. Park, 'GARET: improving throughput using gas consumption-aware relocation in Ethereum sharding environments', *Cluster Computing*, vol. 1, 2020, ISSN: 15737543. DOI: 10.1007/s10586-020-03087-1. [Online]. Available: https://doi.org/10.1007/s10586-020-03087-1.

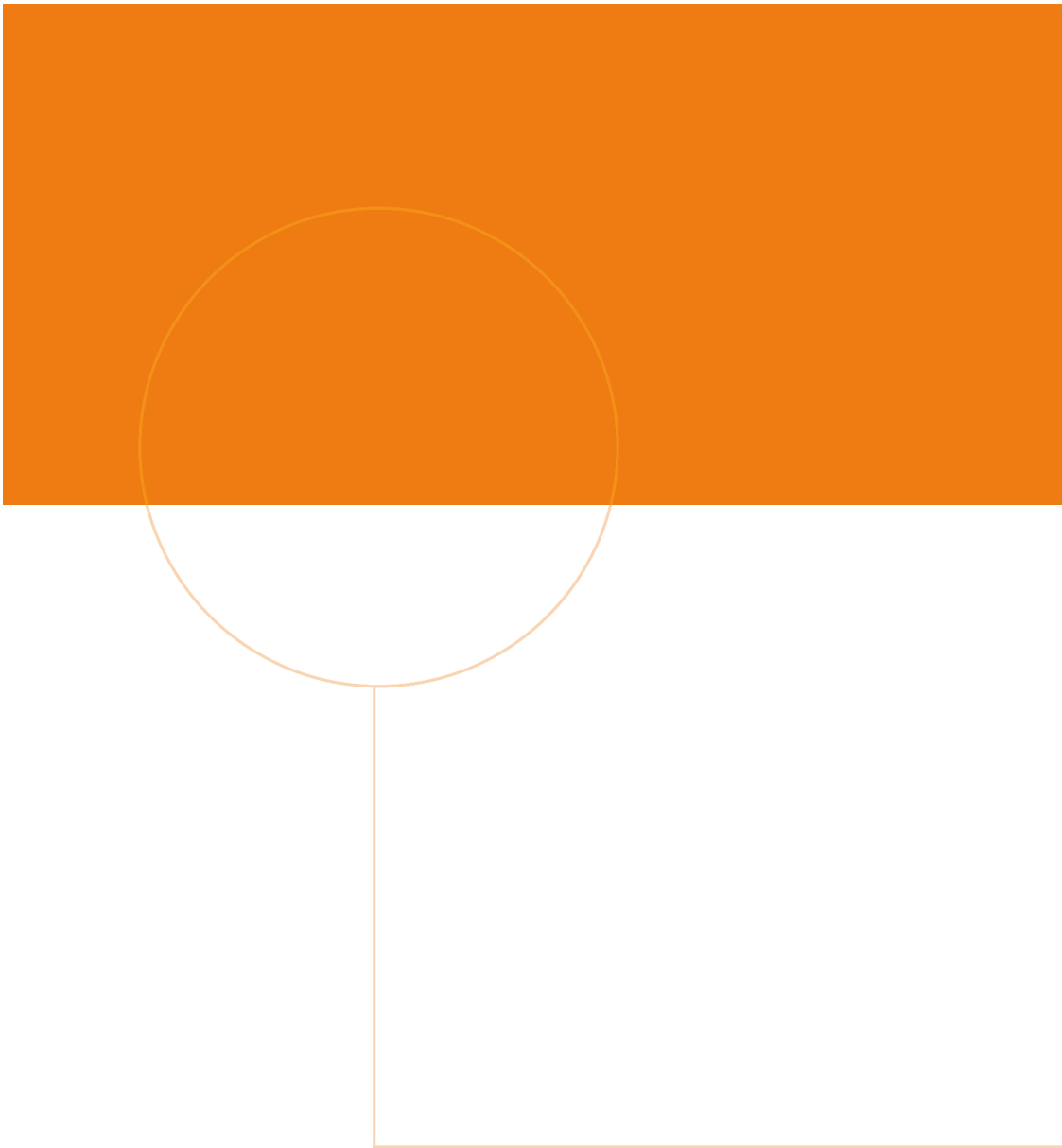
[73] R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze and K. Wehrle, 'How to Securely Prune Bitcoin ' s Blockchain', 2020. arXiv: arXiv:2004.06911v1.

[74] N. Okanami, 'Load Balancing for Sharded Blockchains', pp. 1–13, 2020.

[75] Y. Xu, Y. Huang, J. Shao and G. Theodorakopoulos, 'Multichain-MWPoW: A $p/2$ Adversary Power Resistant Blockchain Sharding Approach to a Decentralised Autonomous Organisation Architecture', 2020. arXiv: 2004.04798. [Online]. Available: http://arxiv.org/abs/2004.04798.

[76] T. Rajab, M. H. Manshaei, M. Dakhilalian, M. Jadliwala and M. A. Rahman, 'On the Feasibility of Sybil Attacks in Shard-Based Permissionless Blockchains', 2020. arXiv: 2002.06531. [Online]. Available: http://arxiv.org/abs/2002.06531.

[77] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang and R. P. Liu, 'Survey: Sharding in Blockchains', *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2965147.



NTNU

Norwegian University of
Science and Technology