Towards Solving the Data Availability Problem for Sharded Ethereum

Daniel Sel[1,2], Kaiwen Zhang[1,2,3,4], Hans-Arno Jacobsen[1,2,3]
[1]Technical University of Munich
[2]Middleware Systems Research Group
[3]University of Toronto
[4]École de technologie supérieure

ABSTRACT

The success and growing popularity of blockchain technology has lead to a significant increase in load on popular permissionless blockchains such as Ethereum. With the current design, these blockchain systems do not scale with additional nodes since every node executes every transaction. Further efforts are therefore necessary to develop scalable permissionless blockchain systems.

In this paper, we provide an aggregated overview of the current research on the Ethereum blockchain towards solving the scalability challenge. We focus on the concept of sharding, which aims to break the restriction of every participant being required to execute every transaction and store the entire state. This concept however introduces new complexities in the form of stateless clients, which leads to a new challenge: how to guarantee that critical data is published and stays available for as long as it is relevant.

We present an approach towards solving the data availability problem (DAP) that leverages synergy effects by reusing the validators from Casper. We then propose two distinct approaches for reliable collation proposal, state transition, and state verification in shard chains. One approach is based on verification by committees of Casper validators that execute transactions in proposed blocks using witness data provided by executors. The other approach relies on a proof of execution provided by the executor proposing the block and a challenge game, where other executors verify the proof. Both concepts rely on executors for long-term storage of shard chain state.

CCS CONCEPTS

• **Information systems** → **Distributed storage**; • **Security and privacy** → **Distributed systems security**; • **Theory of computation** → **Distributed computing models**; **Self-organization**; • **Computer systems organization** → **Peer-to-peer architectures**; *Grid computing*; • **Software and its engineering** → **Ultra-large-scale systems**; **Distributed systems organizing principles**; *Grid computing*; *Software performance*; *Software fault tolerance*; *Software safety*;

KEYWORDS

Blockchain, Ethereum, Sharding, Data Availability, Proof of Stake, Distributed Systems

ACM Reference Format:
Daniel Sel[1,2], Kaiwen Zhang[1,2,3,4], Hans-Arno Jacobsen[1,2,3]. 2018. Towards Solving the Data Availability Problem for Sharded Ethereum. In *2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL'18), December 10–14, 2018, Rennes, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3284764.3284769

SERIAL'18, December 10–14, 2018, Rennes, France

ACM ISBN 978-1-4503-6110-1/18/12…$15.00
https://doi.org/10.1145/3284764.3284769

1 INTRODUCTION

With the arrival of Bitcoin, blockchain technology has consecutively disrupted or impacted numerous industries, institutions and aspects of society. Since Bitcoin enabled simple, fast, and inexpensive money transfer between two parties without trusting any intermediaries, several projects have aimed to extend the concept of blockchain [17]. One of these projects is Ethereum, which introduced the possibility of executing arbitrary code on the blockchain in the form of smart contracts. This extension can allow significantly more complex interactions between multiple parties in a completely trustless environment [2].

However, all of the currently popular and proven public blockchains have one considerable weakness in common: scalability. Increasing the number of nodes or computational power in the network results in an improvement of security, but since every node has to process every transaction, more resources do not result in higher performance [17, 18]. As popularity of these blockchains grow, this inability to scale creates a significant bottleneck, driving up transaction costs and significantly reducing the overall usability of the technology. A recent example that clearly showed the limitations of Ethereum's current scaling capabilities was CryptoKitties, a single distributed application (DAPP) that created enough load to cause a surge in transaction fees and consequently stall a significant amount of user transactions while rendering the platform unusable or unprofitable for numerous other projects.

In the following sections, we will build towards a specific approach of significantly improving the performance and scalability of the Ethereum blockchain and explore current research towards a solution of an inherent challenge in this kind of approach: the Data Availability Problem (DAP).

The contributions of this paper are:

(1) We provide a structured overview of ongoing research on the sharding concept in Ethereum.
(2) We structured recent ideas to an approach to a solution for the DAP.

(3) We propose our design for state transitions in shards that works in synergy with the approach for solving data availability.

The rest of this paper is divided as follows: Sections 2 and 3 provide the necessary background and context to understand the Data Availability Problem presented in Section 4. Section 5 then describes our approach based on the current research in Ethereum while Section 6 summarizes the presented elements and provides an outlook for the future.

2 BACKGROUND ON ETHEREUM

In order to provide the capability of executing arbitrary code in form of smart contracts on the blockchain, Ethereum is constructed out of several core differences to the Bitcoin blockchain [18]:

- The entire state of the network is held in the *state trie*, a Patricia Merkle Trie [9], with the root of the current state included in each block header. This allows nodes to just verify the block headers until the current block and then fetch the required portions of the state from other nodes while verifying them against the current state trie root hash. Executing every transaction since the genesis block as in Bitcoin is not necessary.
- State transitions are executed by the Ethereum Virtual Machine (EVM), which can handle complex arbitrary computations in form of Ethereum bytecode, therefore providing the capability required for smart contracts. The code, balance and current state of every account, including smart contracts, is contained in the state trie.
- Transaction fees consist of two parameters: gas limit and gas price. Each opcode executed by the EVM has a specific cost in form of gas units. Upon submitting a transaction, whether it is just a currency transfer or a smart contract interaction, users are required to specify an upper limit for the units of gas that can be consumed by the action as well as how much Ether (ETH)[1] they are willing to pay per unit of gas.

In addition to executing transactions via the EVM, miners participate in a Proof-of-Work consensus protocol similar to Bitcoin. However, for a successful mining operation, a miner is implicitly forced to store the entire state trie, requiring every miner to also be a full node [18]. This concept does not bear the weakness of the DAP that will be described in the following sections, but suffers from an inability to scale.

For the sake of this paper, we define scalability as the ability to process $O(n)$ transactions, when each network participant only has $O(c)$ resources available, with $n > c$. In this work we focus on quadratic scalability, providing us with the constraint of $n \leq c^2$ [7].

3 CURRENT RESEARCH ON SHARDING

On the protocol layer (Layer 1), Ethereum research is currently focused on a concept named *sharding* as a solution to the scalability challenge. Sharding is the approach of dividing the blockchain into multiple parallel chains with each chain maintaining a security level approximately equivalent to the current single chain. This approach will be presented in more detail in Section 3.2, while the following section provides an excursion into Proof of Stake for better understanding of the background model for sharding [7].

3.1 Proof-of-Stake

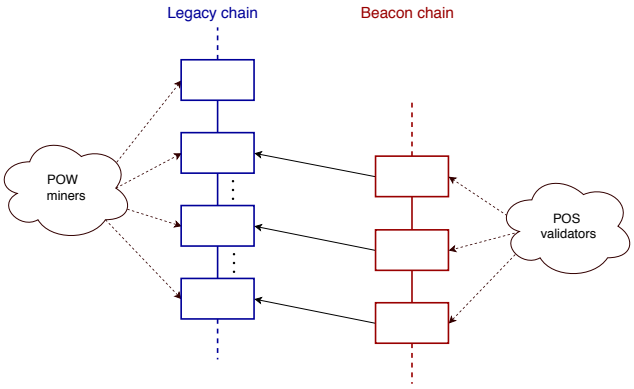

Figure 1: Casper FFG beacon chain as finalization overlay to the PoW legacy chain

One of the major upcoming changes in the Ethereum roadmap is a multi-stage transition from Proof-of-Work (PoW) to Proof-of-Stake (PoS). In the first stage[2], the consensus mechanism will transition into a hybrid PoW/PoS implementation called *Casper*. In Casper, the current blockchain is renamed to *legacy chain*. The legacy chain stays structurally unchanged and continues to use the PoW algorithm for consensus. However, the definition of *finality* is introduced. Blocks on the legacy chain are now interpreted as proposals for blocks on the new *beacon chain*, which achieves consensus on a final and unchangeable history of blocks using the Casper FFG PoS algorithm. [16]. A visual representation of the relationship between legacy chain and beacon chain is shown in Figure 1.

This finalization is achieved using so-called *validators*, which vote on new blocks based on protocol constraints. In order to become a validator, a network participant is required to deposit a significant minimal amount of ETH, the so-called *stake*. As long as the validator participates in the consensus and adheres to the protocol constraints, it will earn rewards proportionally to the deposited stake. Violation of protocol rules leads to slashing (punishing the validator by burning its entire stake) the deposit, while non-participation slowly drains the deposit to avoid deadlocking the network because of unavailable high-stake validators [8].

3.2 Shard-Chains

Simultaneously to the transition from PoW to PoS, Ethereum is planning to implement *sharding* by moving from a single canonical chain to multiple parallel shard-chains. The sharding concept is still under heavy development and in addition to the functional requirements, mainly quadratic scalability, aims to fulfill several objectives, such as achieving the same overall security as a single chain and minimizing the protocol overhead. Since PoS already requires collateralized validators, a reasonable approach is to reuse the Casper

[1] The currency on the Ethereum blockchain.

[2] The second stage is a transition to pure PoS

validators for consensus on the shards. The protocol therefore samples subsets of validators for attesting to shard-blocks (validating through voting), so-called *collations*, for each shard. These subsets are called committees of *notaries* [5]. Additionally, these committees are also responsible for producing checkpoint blocks, called *cross-links*, on the Casper beacon chain that reference collations and therefore tie the shards together with the main chain [5, 6].

However, if committees were fixed and permanently assigned to shards, this would make collusion in a bribing attacker model more likely and since the entire system is as secure as a single shard, it would effectively decrease overall security.

In order to minimize the possibility of collusion, committees should be dynamic with the optimal scenario of random sampling per collation per shard. The challenge hereby is that so far safely validating collations requires knowledge of the state of the shard but validators can not be expected to either keep the state of all the shards or retrieve the entire shard state upon being summoned for a committee since these approaches would not solve the scalability problem.

Therefore, for validators and possibly other roles, a different model applies: the stateless client model [4].

3.3 Stateless Clients

As validators are considered stateless clients in the context of shards, they have to acquire the portions of the state required to verify the collation. Therefore, collation proposers are expected to include *witness* data for every part of the state that is affected by the transactions contained inside the collation body. The included witness data is kept outside of the signed data portion of collations in order to enable notaries[3] to update the witness data in cases where the state changed in the timespan between when a transaction was sent and when it was included in a collation [10].

In the proposed stateless client model, the execution of transactions can be separated from proposing and attesting to collations, effectively yielding three different roles in shards: proposers, executors, and notaries. An analysis of this concept however resulted in the conclusion that in practice proposers are unlikely to exist as a single role [15], which is directing the research to consider merging proposers with executors [14].

At this point it is also not decided how exactly state transitions will be executed in the stateless client concept. The current approaches evolve around executors being long-term associated with shards and therefore storing the state for a specific shard, consequently becoming full nodes in all of their shards. They are also able to provide a proof of independent execution to notaries, effectively freeing them from the responsibility of executing transactions [13].

As in this model, not every participant with critical protocol roles is required or directly incentivized to store and provide data, it leads to a substantial challenge: the Data Availability Problem (DAP).

4 DATA AVAILABILITY PROBLEM

Assuming the stateless client model, where clients are not required to execute every transaction, keep the entire state in storage, or even download collation bodies in order to verify that all collations (and therefore the current state) are valid, two considerable challenges become apparent:

- How can we ensure that the entire data for each collation has ever been published[4]? [1]
- How to guarantee that published data stays available as long as it is possibly relevant for future operations?

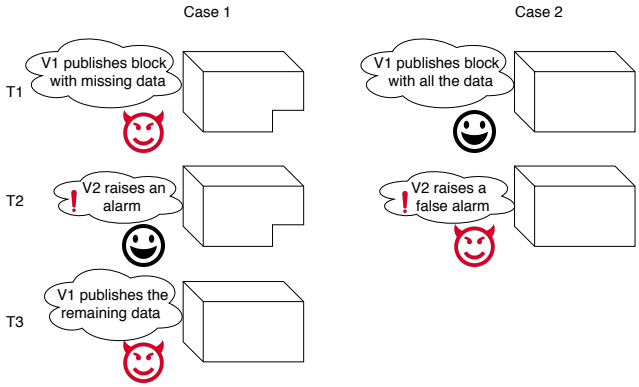

Figure 2: Scenario where unavailable data is not a uniquely attributable fault. Adapted from [3].

A possible approach would be to employ fraud-proofs, where so-called *fishermen* (which could be any participant or a specific role or otherwise restricted group) can, upon discovery of unavailable collations, submit a proof and claim a reward from the slashed deposit of the misbehaving party. Clients could then wait a customizable period of time depending on their risk tolerance if a fraud-proof for a specific state transition or version is published and in the case of absence of fraud-proofs, accept it as valid [3].

However, there is a significant problem with this approach resulting from the fact that unavailable data is not a uniquely attributable fault. In the described scenario, it is impossible for an observer joining at T3 in Figure 2 to determine if the data publisher or the fisherman is acting maliciously.

This leads to an impossible decision regarding the incentive model for fisherman in the described scenario as shown in Figure 3:

- A positive net reward in that situation would allow malicious fishermen to *forge* ETH through false claims.
- A neutral net outcome leads to a Denial-of-Service (DoS) attack vector, where an attacker can force everyone to download every collation body, effectively nullifying the benefits of stateless clients and sharding.
- Punishing the fisherman would lead to only altruistic fishermen reporting fraud-proofs and enables attackers to economically outlast these fishermen and consequently gain the ability to produce unavailable collations.

This constellation leads to some fundamental research questions on the topic of data availability:

(1) Which data has to be published at which time?

[3]As a reminder, validators are called notaries when they act as part of a sampled committee on a specific shard to vote on a specific collation.

[4]With regard to technologies such as zk-SNARKS it is possible to prove and verify the validity of collations without ever requiring the actual data.

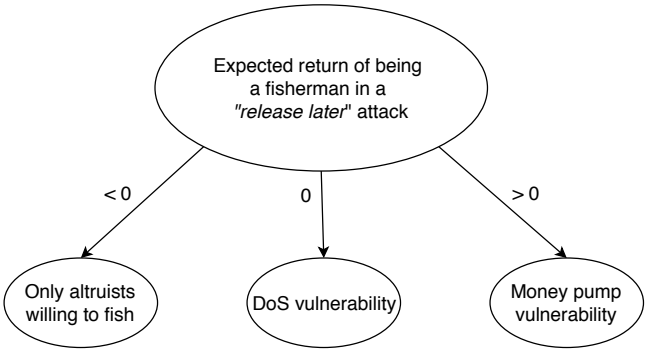

Figure 3: Impossible incentive model for fishermen. Adapted from [3].

(2) Who is responsible in verifying the availability of published data?
(3) Who is responsible for storing which data at which point in time?
(4) How are these responsibilities verified and misbehaviour reliably punished?

As these issues are directly related to critical, consensus-related data, a probabilistic approach alone is not sufficient.

5 CURRENT APPROACH

Bringing all of the moving parts together, we will describe the current approach of solving the DAP in the sharding context of the Ethereum blockchain. As the topic is still in the state of bleeding edge research, this section represents more of a structured collection of rough ideas than a formalized approach. We divide this section into two parts with the first part describing a high-level view of the validation process providing temporary data availability and the second part presenting proposals of integrating state execution in shards and long-term state storage. While the first part is our structured overview based on unorganized research results from the Ethereum Foundation, the second part presents ideas developed by the authors of this paper.

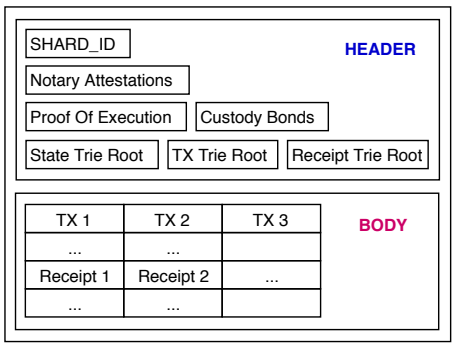

Figure 4: Example for a complete collation

5.1 Validation and Temporary Data Availability

To present a high-level overview of the validation process, we assume that collations are proposed in each shard while skipping the specifics of the collation creation process.

At the beginning of each PoS round[5], a committee of validators is randomly sampled from the pool of Casper validators for each shard using RANDAO as a randomness source [5, 12]. This committee is then responsible for downloading the entire body of the proposed collation (an example is shown in Figure 4) from the entity that created the proposal as well as a fixed number of parent collations, called the *windback*. Every member of the committee then generates a 1-bit custody proof following the process developed by Justin Drake [11]:

(1) The validator generates a secret and publishes the hash of this secret.
(2) The validator splits the data into chunks and XORs them with the secret and the hash of the secret.
(3) The validator computes a Merkle Trie from the resulting chunks.
(4) The validator publishes the hash of the data together with the least significant bit of the resulting Merkle Trie root.

The resulting 1-bit custody bond is then broadcast together with the validator's attestation to the collation. Since a shard collation can refer to a beacon chain block using it's hash in the collation header and vice versa, a single attestation is able to count as a validator's vote for a shard and on the beacon chain simultaneously. As soon as a sufficiently large committee has attested to a beacon chain block referencing a shard block, this reference becomes a cross-link, which tightly couples the state of the shard at the referenced block to the beacon chain and makes a reorganization of that shard impossible after the respective beacon chain block is finalized [16].

After waiting a period of p_s, the validator is required to generate a new secret and reveal the old secret used to generate the custody bond. At this point, every participant in the network can verify and challenge all custody bonds published by that validator during the respective timeframe. Therefore, the data availability is economically incentivized until the challenge period p_c, $p_c > p_s$ is over, which provides us with the desired guarantee for temporary data availability [11].

5.2 State Execution and Long Term Data Availability

In order to actually produce collations, we introduce a new role: Executors. Executors are long-term assigned to specific shards (and might or might not be reshuffled) and responsible for collecting and executing transactions from the transaction pool. Using the same approach as for committee sampling, every round for each shard a proposer, called Executor-Proposer (EP) is chosen out of the respective shard-local executor set. This EP has the privilege and actual responsibility for correctly creating a new valid collection and providing it to the committee of validators for inclusion into the blockchain.

[5] also called timeslot in related research proposals

For the bridging step between execution of transactions and validation of the collation, we propose two reasonable variations, which will be evaluated in more detail in the near future:

(1) Validators verify the state transition by executing transactions with the help of witness data.
(2) EP's generate a *proof of execution*, which is trusted by validators if it is generated by the correct executor. Other executors bear the responsibility of verifying the proof and challenging incorrect results.

5.2.1 Witness Data. As described in Section 3.3, validators are light clients with respect to shards and therefore lack the state data required to execute transactions and fully verify their validity and correctness. One possibility to mitigate this limitation would be for executors to supply the committees with witness data attached to the proposed collation consisting out of the merkle branches in the state trie that are affected by the transactions in the collation.

Members of the committee could then verify correct state execution, which provides the advantage of highly reliable verification of state transitions because of the high stakes involved with Casper validators. Furthermore, this approach would allow for relatively lax policies for executors even up to removing the need for collateralizing them. However, the rules for executors would heavily depend on the desired guarantees for long-term data availability (e.g. since uncollateralized executors who are free to choose shards might disappear and leave shards empty in some cases). Additionally, verifying every transaction by forcing every committee member to execute it limits the overall scalability of the system since the limited set of validators has to provide the computational resources for covering the execution of every transaction in the entire network. This limitation could be partly overcome by distributing transactions between commitee members instead of requiring every member to execute every transaction.

5.2.2 Proof of Execution. As an alternative to requiring the committee to execute transactions, the respective EP could provide a *proof of exection*. The proof of execution is created using the same approach as for the custody bonds described in Section 5.1 while substituting collation data with execution traces from every transaction. An example for a collation with a proof of execution can be seen in Figure 4. Validators would then trust this proof while the EP is guaranteeing for the correctness and validity of the transaction in the proposed collation using a deposit as collateral. Just as in the case of custody bonds, other participants are responsible for verifying and challenging execution proofs as soon as the respective secret is revealed by the EP. In this case, other executors are naturally incentivized to verify the work of the EP since the verification step comes at marginal extra cost for them. The process is visualized in Figure 5.

The advantages of this variant are a reduced load on the committee members, smaller packages since no witness data is required and an extra incentive for executors who are not proposers in the current round. However, these properties come at the cost of considerably reduced flexibility in designing the constraints for the executor role as well as significantly increased complexity in terms of reorganizations of shard chains in case invalid transaction are detected.

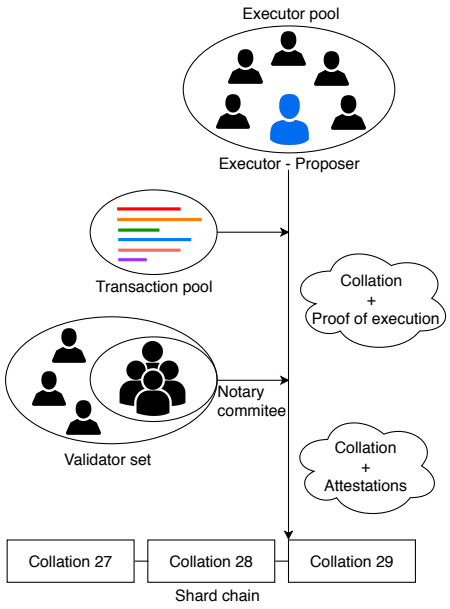

Figure 5: State transition and collation proposal

6 CONCLUSIONS

The research on the sharding concept in Ethereum is at an advanced stage. Numerous new techniques have been developed or established technologies adapted in order to solve unprecedented challenges in scaling a complex blockchain that is able to execute arbitrary code in the form of smart contracts. A major challenge, the Data Availability Problem, which is especially hard to solve because of its property of preventing unique fault attribution, is close to a solution. We propose a sound approach for handling state transitions and validations in a sharded blockchain while guaranteeing data availability. However, this approach has yet to be proven and the entire concept is still bleeding-edge research with open questions.

ACKNOWLEDGMENTS

We thank Vitalik Buterin and Justin Drake for insightful discussions on Ethereum and the DAP. This research is supported by the Alexander von Humboldt foundation. Icons included in our diagrams are made by Freepik and Icomoon from www.flaticon.com.

REFERENCES

[1] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Scalable, transparent, and post-quantum secure computational integrity. *Cryptol. ePrint Arch., Tech. Rep* 46 (2018), 2018.
[2] Vitalik Buterin. 2016. Ethereum white paper: A next-generation smart contract and decentralized application platform. First version (2014).
[3] Vitalik Buterin. 2017. A note on data availability and erasure coding. Ethereum Research Wiki on GitHub. https://github.com/ethereum/research/wiki/A-note-on-data-availability-and-erasure-coding
[4] Vitalik Buterin. 2017. The Stateless Client Concept. Ethereum Research Forum. https://ethresear.ch/t/the-stateless-client-concept/172
[5] Vitalik Buterin. 2018. Attestation committee based full PoS chains. Ethereum Research Forum. https://ethresear.ch/t/attestation-committee-based-full-pos-chains/2259
[6] Vitalik Buterin. 2018. Cross-links between main chain and shards. Ethereum Research Forum. https://ethresear.ch/t/

cross-links-between-main-chain-and-shards/1860
[7] Vitalik Buterin. 2018. Sharding FAQs. Ethereum Research Wiki on GitHub. https://github.com/ethereum/wiki/wiki/Sharding-FAQs
[8] Vitalik Buterin and Virgil Griffith. 2017. Casper the Friendly Finality Gadget. *CoRR* abs/1710.09437 (2017). arXiv:1710.09437 http://arxiv.org/abs/1710.09437
[9] John Byers, Jeffrey Considine, and Michael Mitzenmacher. 2002. *Fast approximate reconciliation of set differences.* Technical Report. Boston University Computer Science Department.
[10] Justin Drake. 2017. Account abstraction, miner data and auto-updating witnesses. Ethereum Research Forum. https://ethresear.ch/t/account-abstraction-miner-data-and-auto-updating-witnesses/332
[11] Justin Drake. 2018. 1-bit aggregation-friendly custody bonds. Ethereum Research Forum. https://ethresear.ch/t/1-bit-aggregation-friendly-custody-bonds/2236
[12] Justin Drake. 2018. Fork-free RANDAO. Ethereum Research Forum. https://ethresear.ch/t/fork-free-randao/1835
[13] Justin Drake. 2018. Proof of independent execution. Ethereum Research Forum. https://ethresear.ch/t/proof-of-independent-execution/1988
[14] Justin Drake and Vitalik Buterin. 2018. Expanding on proposer/notary separation. Ethereum Research Forum. https://ethresear.ch/t/expanding-on-proposer-notary-separation/1691/11
[15] Ben Edgington. 2018. Exploring the proposer/collator split. Ethereum Research Forum. https://ethresear.ch/t/exploring-the-proposer-collator-split/1632
[16] Vitalik Buterin et al. 2018. Casper + Sharding Chain v2.1. Ethereum Research Notes. https://notes.ethereum.org/SCIg8AH5SA-O4C1G1LYZHQ#
[17] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[18] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.