2023

Blockchain security and applications

Phuc D. Thai

BLOCKCHAIN SECURITY AND APPLICATIONS

A submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

PHUC THAI

Bachelor of Science, Vietnam National University, 2017

Directors: Thang Dinh, Hong-Sheng Zhou,

Associate Professors, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

May, 2023

Acknowledgements

I would like to express my gratitude to my advisors, Dr. Thang Dinh and Dr. Hong-Sheng Zhou, for their unparalleled support and guidance. Looking back, I appreciate every rewarding discussion that could last for the whole day. I have learned a great deal from their research expertise and from the mistakes that I made and confronted in those discussions.

Second, I sincerely thank Dr. Tamer Nadeem, Dr. Zhifang Wang, and Dr. Foteini Baldimtsi for serving on my dissertation committee and for providing valuable time and feedback.

I would also like to thank all collaborators at other institutions and my colleagues for their help and support.

Finally, I express my gratitude to my family and friends for their immense support, motivation, and trust in me throughout my academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

BLOCKCHAIN SECURITY AND APPLICATIONS

By Phuc Thai

A submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2023.

Director: Thang Dinh, Hong-Sheng Zhou,

Associate Professors, Department of Computer Science

Cryptocurrencies, such as Bitcoin and Ethereum, have proven to be highly successful. In a cryptocurrency system, transactions and ownership data are stored digitally in a ledger that uses blockchain technology. This technology has the potential to revolutionize the future of financial transactions and decentralized applications. Blockchains have a layered architecture that enables their unique method of authenticating transactions. In this research, we examine three layers, each with its own distinct functionality: the network layer, consensus layer, and application layer. The network layer is responsible for exchanging data via a peer-to-peer (P2P) network. In this work, we present a practical yet secure network design. We also study the security and performance of the network and how it affects the overall security and performance of blockchain systems. The consensus layer is in charge of generating and ordering the blocks, as well as guaranteeing that everyone agrees. We study the existing Proof-of-stake (PoS) protocols, which follow a single-extension design framework. We present an impossibility result showing that those single-extension protocols cannot achieve standard security properties (e.g., common prefix) and the

best possible unpredictability if the honest players control less than 73% stake. To overcome this, we propose a new multi-extension design framework. The application layer consists of programs (e.g., smart contracts) that users can use to build decentralized applications. We construct a protocol on the application layer to enhance the security of federated learning.

CHAPTER 1

INTRODUCTION

The arrival of Bitcoin [82] marked the beginning of the blockchain era, promising a new decentralized economy without the risk of a single point of failure, monopoly, or censorship. In a blockchain system, transactions are recorded and stored in blocks, which are linked together in a chain-like structure (see Figure 1 for an example), hence the name "blockchain". Each block contains a set of transactions and a pointer (hash value) to the previous block. Once a block is added to the chain, it cannot be altered or deleted without invalidating the entire chain, making the system highly secure.

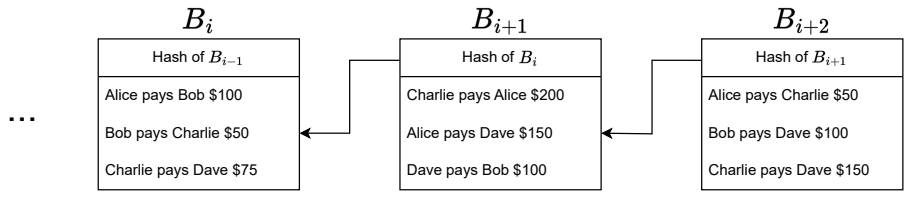


Figure 1: An example of blockchain.

The blockchain protocol provides a secure and public ledger that enables users to view and submit transactions. This public ledger is immutable and tamper-proof, meaning that once a transaction is recorded, it cannot be altered. To achieve this, the public ledger must satisfy two essential properties [88]: *persistency* and *liveness*.

In more details, *persistency* ensures that data stored on the blockchain remains unchangeable, thereby guaranteeing the integrity and security of transactions. This is achieved through cryptographic algorithms that verify and validate the authenticity and accuracy of each transaction before it is added to the public ledger. The

blockchain's distributed architecture also ensures that each participant in the network has a copy of the public ledger, which acts as a safeguard against any attempts to tamper with the data.

Liveness guarantees that the blockchain can process new transactions from users. This is particularly important for applications that require constant updates and real-time data sharing. By guaranteeing liveness, blockchain networks can provide a reliable and secure infrastructure for a wide range of use cases, from financial transactions to supply chain management and beyond. In addition, liveness can contribute to the overall stability and resilience of the network, as it enables nodes to quickly recover from failures and continue processing transactions without interruption.

Overall, the public ledger provides a secure, decentralized and transparent platform for users to conduct transactions, while maintaining the integrity and security of the data stored on it.

Blockchain technology is best known as the underlying technology behind cryptocurrencies like Bitcoin, but it has many other potential applications. For example, it can enhance the security and trustworthiness of multiparty computation [109] by providing a decentralized and immutable ledger for recording the computation results and ensuring that the computation is performed correctly. Other notable applications of blockchain include decentralized finance [105, 61, 101], voting systems [58, 60], and digital identity verification [102, 98].

Layered architecture. Blockchain technology is commonly depicted as a layered architecture (see Figure 2), with each layer building on the layer beneath it. In this dissertation, we will consider the three-layer architecture as follows.

- The *network layer* defines the rules for how nodes communicate with each other and how transactions are propagated through the network. It is responsible for

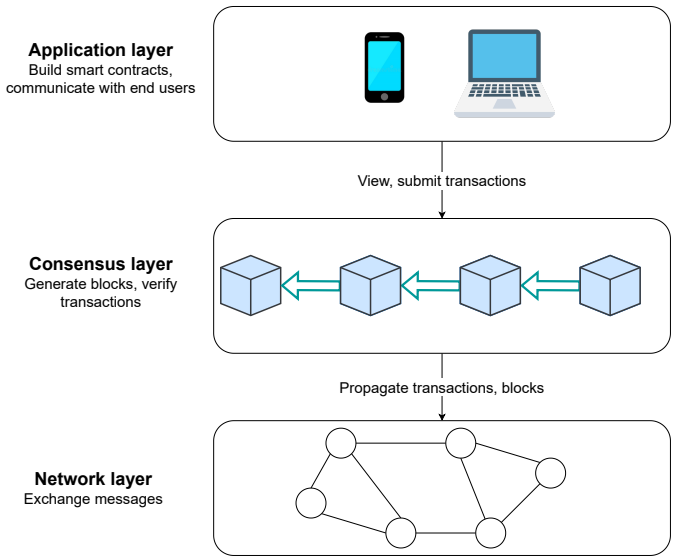


Figure 2: The layered architecture of blockchain.

exchanging messages via a peer-to-peer (P2P) network. For instance, in the Bitcoin network, each node forms an overlay network for propagating messages such as transactions and blocks. To do this, each node establishes 8 outbound connections and accepts a maximum of 125 inbound connections [37, 84]. Once the network is formed, nodes can exchange messages with their neighbors.

- The *consensus layer* is where the rules for validating transactions and creating new blocks are defined. It is in charge of generating the blocks, ordering them, and guaranteeing that everyone agrees. For example, in Bitcoin [82], nodes generate new blocks using a proof-of-work (PoW) mechanism. Specifically, the node that finds *a valid solution* (a random nonce) to *the hash-based PoW puzzle* becomes the block producer for generating the next block. The newly generated block is then sent to the network layer to be propagated to other nodes. Although Bitcoin is one of the most successful cryptocurrencies, its PoW mechanism results in a significant waste of computing resources and

energy. Proof-of-stake (PoS) protocols have been proposed to eliminate the unnecessary waste of energy and computing power in PoW.

- The *application layer* consists of programs (e.g., smart contracts) that can be used by the users to build decentralized applications. The application layer communicates with the underlying layers via application programming interfaces, which allow it to interact with the blockchain network and access data stored on blockchain.

The network and consensus layers offer a blockchain infrastructure that supports building applications on top of it.

1.1 Research Scopes, Objectives, and Motivations of the Dissertation

1.1.1 The network layer

In the network layer is responsible for exchanging messages (e.g., transactions, blocks) that will be used in consensus layer. To ensure security in the consensus layer, the network layer must guarantee that newly generated blocks will be received by *all* nodes *quickly*. We consider the two following properties in the network layer.

- *Reliable dissemination.* The ability that any valid message (broadcasted by an honest node) will be received by all other nodes.
- *Fast synchronization.* The ability of nodes to quickly exchange messages.

The above properties are usually assumed in the existing security analysis [47, 88] for consensus layer.

1.1.1.1 Reliable dissemination

Reliable dissemination guarantees that if any valid message (e.g., blocks, trans-

actions) is broadcasted by an honest node, it will be received by all other nodes within some bounded time. The existing analysis of Bitcoin security [47, 88] assumes the availability of a network functionality that provides reliable dissemination. While this assumption can be achieved in a complete network, it may not hold in real-world P2P networks. These networks are typically sparse, i.e., each node have only a small number of connections. For instance, each node in the Bitcoin network establishes 8 outbound connections and accepts a maximum of 125 inbound connections [37, 84]. Furthermore, nodes maintain the same set of connections over a long period. For example, observations in [14] indicate that 75% of Bitcoin nodes maintain the same set of connections even after 10 hours. To break the reliable dissemination assumption, several attacks have been shown in [56, 103, 78, 50, 86].

Eclipse attack on blockchain's P2P networks. The adversary aims to *eclipse* a node or a group of nodes by *monopolizing* all of their *inbound* and *outbound* connections, effectively isolating the target nodes from the rest of the network. To *monopolize outbound connections*, the adversary can exploit the network protocol to lure honest nodes into establishing more outbound connections toward the adversary-controlled nodes [56]. Furthermore, in the permissionless setting, the adversary can spawn a large number of Sybil nodes, significantly increasing the chances for honest nodes to connect to adversary-controlled nodes. The adversary can *monopolize inbound connections* in Bitcoin network through connection starvation attacks [39, 56]. In this attack, the adversary employs a large number of (sybil) nodes to constantly request to establish connections with the target nodes, filling up all inbound connections of those nodes. Thus, the other honest nodes can hardly connect to the target nodes.

An impossibility against an adaptive adversary. In sparse networks, reliable dissemination is unachievable against an adaptive adversary that can *instantly* corrupt honest nodes and *known the network topology*. Specifically, consider a network with n nodes

and an adversary who can dynamically corrupt any subset of f nodes. In theory, such an adversary can eclipse any node with fewer than $f + 1$ connections by corrupting all of its neighbors [42, 100, 67].

Rigorous designs for reliable dissemination. There has been a growing interest in rigorous designs for reliable dissemination, as shown by recent works [79, 74, 30, 73]. These designs are secure against a (weakly) adaptive adversary (refer to as a mildly adaptive or M-adaptive adversary) that needs to *wait for some time to corrupt* honest nodes.

In [74, 73], nodes randomly establish connections to other nodes based on their resources to construct expander graphs for propagating messages. This prevents the adversary from monopolizing outbound connections. However, these designs lack a mechanism for verifying inbound connections, leaving them vulnerable to DoS attacks by adversaries who can flood honest nodes with inbound connections. To defend against such attacks, nodes can limit the number of inbound connections (similar to the approach used in the Bitcoin network). However, this solution can make the designs vulnerable to connection starvation attacks. Moreover, nodes construct a distinct random graph for each message they send. As a result, over a long period, each node connects to most of the nodes in the network. This contradicts the real-world networks where nodes only have a small number of connections over a long period.

The Generals' Scuttlebutt protocol [30] proposes a new design for reliable dissemination in proof-of-stake settings. In this design, nodes use verifiable random functions to randomly establish connections with other nodes based on their stake. This allows nodes to verify whether other nodes have followed the protocol to establish connections, preventing the adversary from monopolizing both outbound and inbound connections.

Challenges in PoW settings. The existing designs [74, 30, 73] assume that the amount of resource each node controls is known by everyone. This assumption does not hold in PoW settings since there is *no readily available proof of mining power*. Furthermore, binding resources and physical addresses of nodes is unnecessary and can introduce new vectors of attack. An adversary could obtain the addresses of nodes with high connections by leveraging the distribution of resources. These nodes typically control more resources and are critical for network connectivity. The adversary could then perform DoS attacks in an attempt to bring down a small number of these highly connected nodes and disrupt the network. This dissertation proposes a new network protocol to achieve reliable dissemination in PoW settings.

1.1.1.2 Fast synchronization

Fast synchronization refers to the ability of nodes to quickly exchange messages. This offers better security for achieving consensus [88]. Specifically, after a node generates a new block, it must be propagated to the other nodes in the network before they can begin mining on it. This process of block propagation can take some time and cause asynchronization between nodes. As a result, nodes may end up mining on different blocks, which can lead to a waste of mining power. After removing the waste of mining power, we refer to the remaining mining power as "effective mining power". With faster synchronization, nodes can reduce the waste of mining power and increase effective mining power. As shown in [88], if the adversary has more mining power than the effective mining power, it can break the security of the system. Therefore, nodes need to synchronize blocks faster to require the adversary to have more mining power to attack the system.

Heterogeneous bandwidth distribution. One reason for asynchronization in a blockchain network is the highly heterogeneous distribution of bandwidth among

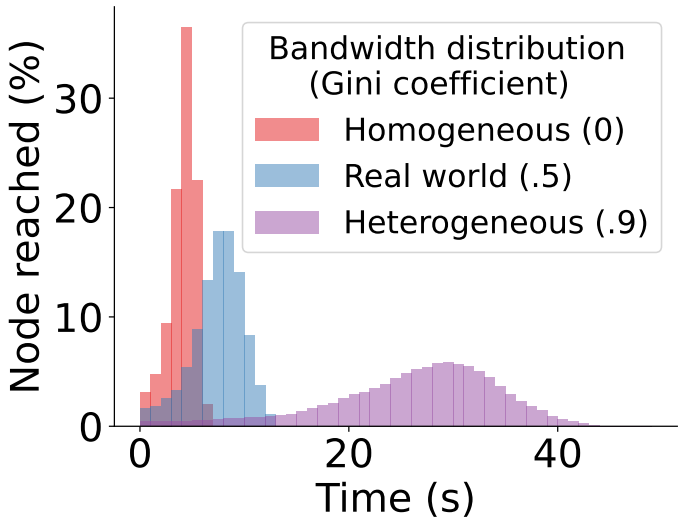


Figure 3: The block propagation time of Bitcoin.

nodes. For example, a study of IPv4-based Bitcoin nodes in [49] found an average bandwidth of 73.1 Mb/s, with a Gini coefficient of 0.5. Figure 3 illustrates how block propagation time in the Bitcoin network rapidly increases as the distribution of bandwidth becomes more heterogeneous. In this dissertation, we propose a new network protocol that provides fast synchronization, even when the distribution of bandwidth is highly heterogeneous.

1.1.2 The consensus layer

The consensus layer defines the rules for validating transactions and creating new blocks. The protocols in the consensus layer provide a public ledger that allows users on the application layer to view and submit transactions. Once a transaction is recorded on the public ledger, it cannot be altered. To achieve this, the public ledger must satisfy two properties [88]: *persistency* and *liveness*. As shown in Pass et al. [88], the persistency and liveness properties of the public ledger are proven based on three *security properties for consensus* called *chain growth*, *common prefix*, and

chain quality. The chain growth property ensures that the blockchain continues to grow over time. The common prefix property ensures that all nodes in the network have the same view of the blockchain up to a certain point in time. The chain quality property ensures that a significant fraction of blocks are generated by honest miners. We say a consensus protocol is secure if it achieves chain growth, common prefix, and chain quality properties. In this dissertation, we focus on designing a secure proof-of-stake consensus protocol.

Proof-of-stake (PoS). Bitcoin is widely regarded as one of the most successful cryptocurrencies to date, yet its utilization of the Proof-of-Work (PoW) mechanism has led to significant concerns regarding energy consumption and computational waste. PoW requires an enormous amount of computing resources to solving the hash-based PoW puzzle, which in turn leads to high energy usage and environmental impact. Fortunately, proof-of-Stake (PoS) protocols(e.g., [2, 70, 99, 13]) have proposed as an alternative solution. These protocols offer a more energy-efficient and eco-friendly alternative to PoW. In a nutshell, in a PoS based blockchain protocol, players are expected to prove ownership of a certain number of stake (coins); only the players that can provide such proofs are allowed to participate in the process of maintaining the blockchain.

In order to extend the chain, the players make attempts to find the *solutions* to the *the hash-based PoS puzzles*. Note that, the PoS puzzles are defined based on "contexts"; Usually, the context is extracted from the previous blocks on the blockchain. In our protocol, the context is the hash value of the last block on the longest chain. In a very high level, the context could be used as a (biased) randomness to determine which players can generate the next block. The solution to the puzzle is based on the information of the stake, a time step (round number), and the context. Thus, in comparison with PoW mechanisms, the computational cost to find the solutions in

PoS mechanisms is very "cheap".

From ad hoc to rigorous designs. Early PoS designs (e.g., [2, 70, 99, 13]), are in an *ad hoc* style. This is the same for PoW based design: the original Bitcoin design is indeed in an *ad hoc* style. The recent trend is to follow a rigorous approach: security concerns are carefully defined and the designed protocols are mathematically analyzed. Notable efforts include the work in [34, 32, 7] for proposing provably secure PoS protocols.

(Un)predictability. Intuitively, predictability means that (certain) protocol players are aware that they will be selected to generate blocks of blockchain, *before* they actually generate the blocks. Brown-Cohen et al. [19] have studied the (un)predictability of PoS in incentive-driven settings (in which the players will deviate from the protocol if doing so yields higher profit). *The "power" of predictability can be abused by the attackers* so that they can reduce the difficulty/cost of performing many incentive-driven attacks such as selfish-mining [19], or bribing [7]. In a selfish-mining attack, an attacker gains an unfair advantage by selectively withholding or revealing blocks. With predictability, the attacker can develop a better strategy to perform the attack. In a bribery attack, an attacker bribes players to work on specific chains in order to benefit themselves, such as supporting double-spending or censorship attacks. By predicting which players are likely to mine new blocks, the attacker can attempt to bribe them. Such attacks undermine the fairness of the blockchain and discourage honest participation.

Therefore, it is crucial for a PoS protocol to minimize predictability and mitigate the risks of such attacks. Ideally, we expect a PoS protocol to achieve the (best possible) unpredictability property so that the attacks based on the predictability can be addressed *as much as possible*. By achieving this goal, the fairness of the

blockchain can be maintained, and honest players can be incentivized to participate in the protocol. Unfortunately, the state-of-the-art PoS protocol in [34, 32] cannot achieve the best possible unpredictability.

1.1.3 The application layer

The application layer is built on top of the consensus layer. The users of applications can view and submit transactions to the public ledger (provided by the protocol in consensus layer). As the public ledger offers decentralization, transparency, immutability, the applications can provide a high degree of data integrity, and can enforce rules via self-executing smart contracts. In this work, we focus to design an application of blockchain to enhance the security of federated learning (FL).

Federated learning has been developed to simultaneously offer both data privacy and high performances in training models, via a distributed learning protocol. It address the privacy concerns for critical applications of machine learning with sensitive data such as health care [25, 91] and the internet of things [87, 107]. By design, FL allows clients to collaboratively train a global model without having to disclose their private training data. In each training round, a central server distributes the current global model to a random subset of clients who will train locally and upload model updates to the server. Then, the server averages the local updates into a new global model. As training data never leaves the clients' devices, FL is widely regarded for preserving data privacy.

However, the promise of privacy in FL has been repeatedly contested. Prior work has shown that the local model updates do, in fact, leak some sensitive information about the client's private data, thereby making them a vector for privacy attacks from FL servers [96, 95, 48, 44]. To tackle this issue, recent research has focused on concealing the local models via devising secure aggregation protocols [17, 5, 106].

With this new framework, the central server can compute the average of the local models to update the global model in a way that each individual local model is kept private. As a result, the local model updates are protected from the server, thereby preventing the server from exploiting the updates of any client for privacy attacks. However, the FL server can manipulate the client selection process to circumvent the secure aggregation protocols. The server can then obtain the local model updates of a targeted client, and exploit those updates to compromise their data privacy. In this dissertation, we leverage the properties of blockchain to design a secure client selection protocol.

1.2 Contributions of the Dissertation

This dissertation focuses on designing secure blockchain protocols for network and consensus layers, and leveraging the security of blockchain to develop applications.

1.2.1 The network layer

1.2.1.1 Reliable dissemination

We propose a novel design, called CoSpaN, for reliable dissemination in PoW settings. Here, CoSpaN stands for **Co**nsensus in **Spa**rse **N**etworks.

Proof of mining power. Nodes use *merge-mining* to provide proofs of mining power, using the same PoW for generating new blocks, albeit with different thresholds. This way, the number of proofs of mining power is proportional to the mining power of nodes. After generating proofs of mining power, nodes submit them to the blockchain to make them readily available to everyone.

Based on proofs on mining power, nodes construct a bi-level random graph,

called a *core-periphery graph*. Each proof of mining power entitles a node to act as a *core node* and establish connections to other nodes. Nodes that are not associated with any core nodes are referred to as *periphery nodes*. The core nodes select a set of neighbors, which includes both core and periphery nodes, to establish outbound connections. The selection of neighbors depends on the nodes' information and a random token extracted from the blockchain.

To ensure the connections are random and verifiable by the participating nodes of each connection, we design the functions to select the set of neighbors using inequalities over verifiable random functions (VRF) [40]. In addition, using VRF also ensures the confidentiality of connections, i.e., each connection is known only by the two participating nodes. This prevents adversaries from learning the network topology and allows the protocol to achieve reliable dissemination in a sparse network against an adaptive adversary. Shielding the network topology also provides protection against network disruption attacks such as BGP hijacks [94].

DoS resilience. To defend against DoS attacks on highly connected nodes, CoSpaN protocol preserves the confidentiality of core nodes by hiding the association between core nodes and their physical addresses. As a result the adversary cannot determine whether or not a node is associated with a core node. This prevents the adversary from launching DoS attacks to bring down core nodes and disrupt the network.

Security analysis in different threat models. Besides proving the security of CoSpaN protocol against the M-adaptive adversary, we also prove that the CoSpaN protocol is secure against a (weakly) *slow-observation* adaptive adversary (S-adaptive). In contrast with the M-adaptive adversary, the S-adaptive can *instantly corrupt* honest nodes but need to *wait for some time to discover a newly established connection*.

Comparision to existing designs. We compare the proposed CoSpaN protocol

Protocols	PoW	DoS resilience	Reliable dissemination	Verifiable random connection	Sparse network	Non-mining nodes	Security analysis for consensus
[74]	✗	✗	✓	✗	✗	✗	✗
[73]	✗	✗	✓	✗	✗	✓	✗
[30]	✗	✗	✗	✓	✓	✗	✗
CoSpaN	✓	✓	✓	✓	✓	✓	✓

Table 1.: A comparison between CoSpaN protocol and existing designs for reliable dissemination.

with existing designs for reliable dissemination. Please refer to Table 1 for a summary.

PoW settings. The existing designs in [74, 30, 73] cannot be directly applied in PoW settings because there is no readily available proof of mining power. CoSpaN is the first design to provide reliable dissemination for sparse networks in PoW settings.

DoS resilience. In the designs in [74, 30, 73], the adversary could obtain the addresses of nodes with high connections by leveraging the distribution of resources and then perform DoS attacks on those nodes.

Reliable dissemination. The Generals' Scuttlebutt protocol [30] can only guarantee *partial* reliable dissemination. This means that a message broadcasted by an honest node will only be received by a large fraction of nodes in the network. In contrast, CoSpaN and the designs in [74, 73] can achieve reliable dissemination.

Verifiable random connections. The designs in [74, 73] lack a mechanism for verifying inbound connections, allowing the adversary to monopolizing the inbound connections of nodes. Meanwhile, CoSpaN and Generals' Scuttlebutt use VRF to allows the participant nodes to verify the connection.

Sparse networks. We say a network is sparse if each nodes in the network only connect with a few other nodes over a long period. In the designs in [74, 73], nodes construct a distinct random graph for each message they send. Thus, over a long period, each node connects to most of the nodes in the network. On the other hand, the network in CoSpaN and Generals' Scuttlebutt protocol are sparse since nodes keep the same

set of a few connections over a long period.

Non-mining nodes. The designs in [74, 30] provide no connectivity to non-mining nodes. Those non-mining nodes, such as relay nodes and nodes that provide the tools and infrastructure for DApps, play an important role in blockchain ecosystems. In contrast, CoSpaN and the design in [73] provide connectivity guarantees for all nodes, including non-mining nodes.

Security analysis for consensus. The designs in [74, 30] completely separate the network design and the consensus design. While this approach simplifies the analysis for network designs, it adds extra complexity to the analysis of the overall protocol (including both network and consensus designs). For example, the network design in [30] can only guarantee partially reliable dissemination. Hence, the existing analysis, which assumes reliable dissemination, cannot be used. The CoSpaN protocol makes minimal modifications to the consensus design. Thus, we can use the existing analysis [47, 88] to analyze the security of our protocol.

1.2.1.2 Fast synchronization

We investigate the theoretical bounds on the fast synchronization in terms of both throughput and latency in a blockchain system, consisting of nodes with *heterogeneous capacities*. We assume that major traffic in blockchain, such as Bitcoin, are due to transaction propagation [85] and traffic due to block propagation becomes negligible due to improvement such as Compact Block Relay [29]. Thus, the data can orignate from any nodes before being broadcasted to all nodes in the P2P network. Similar to previous works [71, 68, 26, 24], we assume the upload-links are the bottlenecks, i.e., nodes are only constrained by their upload capacities (but not their download capacities).

In contrast to common folklore that the throughput is limited by the node with

the weakest capacity, we show a stronger theoretical limit: the average upload capacity of nodes. We formulate the throughput problem as a Blockchain P2P Network Design Problem, asking for what throughput (data arrival rates) there exist a feasible transmission schedule with a low average degree and latency. More importantly, we prove that there exists a transmission schedule for any throughput up to $(1-\epsilon)$ the theoretical limit, the average upload capacity. In addition, the transmission schedule has an average degree $O(1/\epsilon)$ and a latency is optimal up to a factor $O(\log n)$.

More importantly, there exist *sparse* P2P network design that approach *arbitrarily close* to the theoretical limit. Our result implies that nodes with higher capacities can make up for the shortage in upload capacity of weaker nodes, providing a route to strengthen the throughput by adding strong nodes into the P2P network. Our construction implies a practical approach for designing real-world blockchain P2P network. To maximize the throughput, nodes should create *same-capacity* links, i.e., the *number of links should be proportional to the nodes' upload capacity.*

1.2.2 The consensus layer

We identify an interesting impossibility for a class of PoS protocols that follow a *single-extension* design framework. Existing provably secure Bitcoin-like PoS protocols (e.g., [34, 32, 7]) are all in the single-extension framework. To overcome the impossibility, we introduce a new design framework, called *multi-extension*. We develop a novel D-greedy strategy in the multi-extension framework, which allows us to design provably a secure Bitcoin-like PoS protocol. Finally, we present new analysis techniques to analyze the chain growth and the common prefix properties for PoS protocols in multi-extension framework. We next elaborate our results.

Impossibility result of single-extension protocols. We formally define a *single-extension* framework for constructing PoS protocols, which is followed by existing

PoS protocols such as Ouroboros Praos [34], SnowWhite [32], and Bagaria et al. [7]. In a single-extension protocol, each honest player selects a chain using a best chain algorithm and then attempts to extend it as follows. The player first extracts a context from the chain using a context extraction algorithm. Then, the player uses the context, the current round number, and secret key to determine whether or not they are allowed to generate a new block at that round.

We have identified an interesting impossibility result for single-extension PoS protocols: *For any single-extension PoS protocol that achieves the best possible unpredictability, it cannot achieve the common prefix property if the honest players control less than 73% of the stake.*

We prove the impossibility based on a new property that we formulated called *distinct-context-extension* in PoS protocols. The distinct-context-extension property states that the contexts of any two valid chains are different. Our proof consists of two steps as follows. First, we show that if the single-extension PoS protocol achieves the best possible unpredictability, it must have the distinct-context-extension property. Intuitively, if the extension of the two different chains are shared-context-extension (the opposite of the distinct-context-extension property), a player can predict whether or not she/he can extend one chain after making an attempt to extend the other chain. Thus, if the protocol does not achieve the distinct-context-extension property, it cannot achieve the best possible unpredictability. Secondly, we show that if the protocol has the distinct-context-extension property, then it cannot achieve the common prefix property if the honest players control less than 73% stake. We consider an adversary that extends a set of chains privately, and we can bound the chain growth of the adversary by using a random tree to model the chain extension of the adversary. We can show that the adversary can amplify its stake by a factor e, where $e = 2.72$ is the base of the natural logarithm. Therefore, if the honest players control less than

73% stake, the adversary can extend the chain faster than the honest players, thus breaking the common prefix property.

We remark that, we are the first to present the impossibility result for the single-extension protocols. Our previous version and the work in [7] showed that some single-extension protocols can be secure with 73% honest stake. However, those works never claim the impossibility result. To prove the impossibility result, we formally define the single-extension protocol and the new concept of distinct-context-extension property.

New design: Overcoming the impossibility via multi-extension. To overcome the impossibility, we propose a new design framework, called *multi-extension*, for PoS protocols. In a multi-extension protocol, each honest player is allowed to extend multiple chains in a round, rather than just one. We remark that, designing a secure and practical multi-extension PoS protocol is challenging. For example, Bagaria et al. [7] have shown that a protocol allowing for the honest players to extend slightly shorter chains than the best chain is vulnerable to "balance attacks" which can break the common prefix property. Fortunately, we can have a particular design that follows a novel "D-distance-greedy" strategy, that can be proven to be secure.

D-distance-greedy strategy. We propose a novel *D-distance-greedy* strategy that allows the honest player to a set of best chains, where D is a positive integer. By using the D-distance-greedy strategy, we can construct a protocol that achieves the best possible unpredictability. At the same time, the protocol can be proven to be secure with a smaller fraction (e.g., 57%) of honest stake.

In the D-distance-greedy strategy, the players extend a set of best chains that are "close" to the best chain. We say a chain is "close" to the best chain if a common prefix can be obtained by removing the last D blocks from the best chain. This

ensures that the honest players extend a set of chains that share the same prefix, making it impossible for adversary to launch balance attacks.

A new tiebreak rule. In a multi-extension protocol, the probability of generating a new best chain can change depending on the number of chains in the set of best chains. Consider a protocol execution round, and there are two longest chains; different strategies of choosing the best chain may increase or decrease the probability of generating a new one. This creates an opportunity for an adversary to *slow down* chain growth by publishing a chain with the same length but with fewer chains in the set of best chains, as the best chain. To address this issue, we introduce a new tiebreak rule: If there are multiple chains with the same length, then the chain that can be extended the fastest, will be selected as as the best chain.

Intuitively, honest players will generate a new best chain faster if there are more chains in the set of best chains. To make this more concrete, we can partition the set of best chains into $D + 1$ subsets based on their depth. The depth of a chain is the difference between its length and the length of the current best chain. For $i \in [0..D]$, we denote the i-depth subset as the set of chains at depth i. A new best chain is generated if a player extends a chain in the 0-depth subset, which consists of all chains with the same length as the current best chain. Additionally, for $i \in [0..D-1]$, a new chain is added to the i-depth subset if a chain in the $(i+1)$-depth subset is extended. Therefore, we compare the number of chains in each depth-based subset, from the 0-depth subset up to the D-depth subset, to break ties between chains of equal length. This way, players can select the best chain that can be extended the fastest.

New analysis: Chain growth in multi-extension. To analyze the chain growth property, we propose a new Markov chain analysis framework to study the chain

growth in multi-extension protocols. Then, we apply the new analysis framework to analyze the chain growth of our protocol. We consider a *hybrid experiment*, where all messages sent by the adversary are removed. Based on our tiebreak rule, we can show that the chain growth in a real execution is lower bound by the chain growth in the hybrid execution. Note that, the hybrid experiment have been introduced in the analysis in [88] to analyze the chain growth of Bitcoin protocol. In our protocol, the honest players may extend multiple chains in a round. Thus, we design a *random walk* on a *Markov chain* to analyze the chain growth in the hybrid experiment for our protocol.

We start by designing a *simplified Markov chain* and then proceed to design an *augmented Markov chain*. Recall that, the set of best chains can be partitioned into $D + 1$ subsets based on the depth of those chains. A new best chain is generated if a player extends a chain in the 0-depth subset, which consists of all the chains that have the same length as the current best chain. We can analyze chain growth by analyzing the expected number of chains in the 0-depth subset of each round. In the simplified Markov chain, each state represents a protocol round with specific numbers of chains in all depth-based subsets. The transitions on the Markov chain depends on how the set of best chains is updated after each round. The simplified Markov chain only gives information about the depth-based subsets, but it doesn't show how many chains are removed when a new chain is generated. This makes it hard to find a good lower bound for the amplification ratio, even with a large D.

To solve the issue in the simplified Markov chain, we propose an augmented Markov chain. We use *depth-distance-based subsets* that select chains based on both their length and distance from the best chain. The augmented Markov chain shows a more detailed representation of the best chains, so we can identify which chains belong to the new set when a new best chain is generated. This gives a better lower

bound for the amplification ratio.

We remark that, the Markov chain technique has been used to analyze the common prefix property [65] for Bitcoin protocol, which follows a single-extension fashion. However, our Markov chain is very different from the Markov chain in [65]. As our protocol follows the multi-extension framework. Thus, a more complex Markov chain is needed to analyze the chain growth property.

New analysis: Common prefix in multi-extension. Previous analysis of Bitcoin's PoW consensus [47, 88] showed that the key factor for establishing the common prefix property is that honest participants can contribute only one block at a block height. Breaking this property requires the adversary to generate more blocks than the honest participants, which is infeasible due to the majority control of mining power by the honest participants. Our proposed protocol aims to defend against nothing-at-stake attacks by allowing players to extend multiple chains. Thus, we can no longer guarantee that honest participants contribute only one block per block height.

To analyze the common prefix property for the multi-extension protocol, we introduce the notions of *virtual block-sets* and *virtual chains*, and then define the *common prefix property w.r.t. virtual chains*. We can prove the common prefix w.r.t. virtual chains by showing that the honest players only contribute at most one virtual block-set at a block height. Afterward, we show that the standard common prefix property can be reduced to common prefix w.r.t. virtual chains. In detail, a virtual block-set consists of multiple blocks with the same height that are "close" to each other. We first define two chains are "close", and then define two blocks are "close." We say two chains are close if they share a common prefix in a recent past. When two chains are close, the last blocks of the two chains are also "close". We define the virtual chain consists of multiple virtual block-sets that are linked together. Note that, the adver-

sary cannot use the blocks of honest players to break the common prefix property, the adversary needs to generate more virtual block-sets than the honest players to break the common prefix property (w.r.t virtual chains). This requires the adversary to control the majority of the stake (which contradicts the assumption that the honest players control the majority of the stake). Finally, we show the common prefix w.r.t. virtual chains implies the regular common prefix property. This is given by the fact that the blocks in the same virtual block-set are "close".

Best possible unpredictability. Our protocol minimizes predictability by using the hash value of the last block to extract the context. This ensures that the contexts of any two different chains in the execution of our protocol are different, i.e., our protocol achieves distinct-context-extension property. As a result, players cannot predict the extendibility of a future chain based on the extensions of a current chain. Players can only predict whether or not they can extend the current best chain. Our protocol provides the best possible unpredictability for PoS protocols.

Extensions: Full-fledged blockchain and adaptive stake registration. Our protocol can be "upgraded" to a regular blockchain to include payload, such as transactions, in the blocks. The chain in our protocol serves as a randomness beacon to select a PoS player to generate a new block and extend the blockchain. Similar to [6], we also allow new players to join the system and participate in the process of extending chains, as long as they have registered their stake a specified number of rounds earlier. This ensures that the adversary cannot gain any extra advantage.

1.2.3 The application layer

To defend against the client selection attacks, we first formulate a *secure client selection (SCS)* problem in which the goal is to enforce semi-malicious (and malicious) adversaries to select random pools of clients in each training round, thus, providing an

effective defense against the newly proposed client selection attacks. Importantly, we narrow down the defense into achieving three necessary security properties, namely, *pool consistency, pool quality,* and *anti-targeting*. We present a framework, termed **Se**cure **P**rotocols (SeP), for achieving such security properties, leveraging blockchain as a public trust entity for randomness generation and verifiable random function (VRF), a cryptographic tool to make the client selection both random and unpredictable.

Communication-efficient Design. To mitigate the amount of communication, we extend SeP into a **Co**mmunication-efficient and **Se**cure **P**rotocol (CoSeP) for the SCS problem. In CoSeP, we develop *light-weight client communication procedure*, where the clients only need to retrieve the block headers, which are much smaller than the blocks from the blockchain, to verify the correctness of the client selection. Our communication procedure uses a so-called *pass-through communication*, in which the server will gather the information of the selected clients and only submit a logarithmically-small commitment of that data to the blockchain.

Defending against active adversaries. While pass-through communication helps reduce communication complexity, it may introduce new attack vectors for an *active adversary* that can divert from the protocol. For example, the active adversary can manipulate the client selection process by corrupting the server and excluding the information of some clients in the pass-through communication. Hence, we devise protocol CoSeP_+ by adding extra steps to CoSeP which enable the clients to dispute if the server does not include their data in the pass-through communication.

1.3 Organization of the Dissertation

In Chapter 2, we present CoSpaN protocol for reliable dissemination in PoW settings. This chapter is based on:

Phuc Thai, Hong-Sheng Zhou, Lei Fan, Jonathan Katz, and Thang N Dinh. "CoSpaN: Permissionless Consensus in Sparse Networks."

In Chapter 3, we propose propagation scheme for fast synchronization in heterogeneous blockchain networks. This chapter is based on:

Phuc Thai, Hong-Sheng Zhou, My T. Thai, Tam Vu, and Thang N Dinh. "Asynchronization in Heterogeneous Blockchain Networks: Optimization and Mitigation Principles."

In Chapter 4, we present a new PoS consensus protocol with the best possible unpredictability . This chapter is based on:

Phuc Thai, Lei Fan, Jonathan Katz, and Hong-Sheng Zhou. "A Permissionless Proof-of-Stake Blockchain with Best-Possible Unpredictability."

In Chapter 5, we leverage blockchain technology to propose an application to securely select clients in federated learning process. This chapter is based on:

Truc Nguyen, Phuc Thai, Jeter Tre'R, Thang N. Dinh, and My T. Thai. "Blockchain-based Secure Client Selection in Federated Learning."

CHAPTER 2

THE NETWORK LAYER: RELIABLE DISSEMINATION

In this chapter, we present CoSpaN, the first network protocol designed for sparse networks in PoW setting. Nodes use merge-mining to provide proofs of mining power, and establish verifiable random connections such that the expected number of connections for each node is proportional to their mining power. Additionally, the CoSpaN protocol preserves the confidentiality of core nodes. This means that the adversary cannot determine whether or not a node is associated with a core node. As a result, the adversary is unable to launch DoS attacks to bring down core nodes and disrupt the network.

We organize the chapter as follows. In Section 2.1, we present a model for consensus in sparse networks. The design of CoSpaN protocol is presented in Section 2.2. The analysis of the security of CoSpaN against a static adversary and adaptive adversary are presented in Section 2.3 and Section 2.4, respectively. The numerical studies are shown in Section 2.5.

2.1 Consensus Models for Sparse Networks

In this section, we introduce our network-aware consensus model, which captures the *sparse* nature of real-world P2P networks. We further define the formal concept of *reliable dissemination* and the task of designing *consensus in sparse networks*, providing the first theoretical framework for analyzing consensus protocols in sparse networks.

2.1.1 Sparse network model (SNM)

We introduce a *sparse network model* (SNM) for a blockchain protocol execution. Extending the model by Pass, Seeman, and Shelat [88], referred to as the PSS model, our model includes a set N that consists of n ($n \in \mathbb{N}$) participating nodes, a blockchain protocol (Π, Λ), an environment $\mathcal{Z}$ that captures all aspects external to the protocol itself, and an adversary $\mathcal{A}$. The execution proceeds in *rounds* that model time steps.

There are three major differences between the SNM model and the PSS model. First, we consider a *point-to-point communication* in which nodes can only send messages after establishing connections to other nodes. This contrasts with the PSS model, which considers a communication model in which nodes can send messages to all other nodes. Secondly, we consider a *non-flat* mining model where each node can have a different amount of mining power, in contrast to the flat mining model (i.e., all nodes have the same mining power) in PSS. In particular, our non-flat model allows *nodes with zero-mining power*, e.g., nodes that participate in relaying messages but not the mining process. Such nodes and their corruption were not investigated in the PSS model. We are the first to *go beyond the standard corruption of the mining nodes and investigate the corruption of zero-mining power nodes that are responsible for communication functionality.* Finally, in the SNM model, we start with a *static adversary* that controls the same subset of nodes during protocol execution. Then, in Section 2.4, we consider two (weakly) *adaptive adversaries* who may corrupt honest nodes during the protocol execution but in certain restricted manners. [1]

Participating nodes. The set of participating nodes N can be partitioned into

[1] In the PSS model, nodes (with mining power) are allowed to be corrupted in an adaptive manner; furthermore, corrupted nodes can be uncorrupted. In our SNM model, both nodes with mining power and the ones with zero-mining power are allowed to be corrupted in weakly adaptive manners; in addition, already corrupted nodes are *not* allowed to be uncorrupted.

the set of *honest nodes* $\mathsf{H} \subset \mathsf{N}$, who strictly follow the blockchain protocol (Π, Λ), and the set of *malicious nodes* $\mathsf{N} \setminus \mathsf{H}$, which are controlled by an *adversary* $\mathcal{A}$. The number of malicious nodes $f = |\mathsf{N} \setminus \mathsf{H}|$ is bounded by $\eta \cdot n$ for a fixed $\eta \in (0, 1)$.

A blockchain protocol. Algorithm Π, which takes a security parameter κ as input, dictates how each node communicates and maintains a blockchain data structure $\mathcal{C}$, a collection of blocks. Algorithm Λ contains a set of rules to extract a ledger $\mathcal{L}$ from $\mathcal{C}$, i.e., $\mathcal{L} = \Lambda(\mathcal{C})$. Here, the ledger $\mathcal{L}$ is a sequence of transactions, i.e., $\mathcal{L} = tx_1 \| tx_2 \| \ldots \| tx_i \| \ldots$. We also overload the notation Λ to define the position of a transaction tx in the ledger. Specifically, for a transaction tx, $\Lambda(\mathcal{C}, tx) = i$ if $tx = tx_i$, the i-th transaction in the ledger, and $\Lambda(\mathcal{C}, tx) = 0$ if tx is not included in the ledger. The validity of a ledger is captured by a predicate V. The predicate $\mathsf{V}(\mathcal{L})$ returns 1 if and only if the ledger $\mathcal{L}$ is *valid* for some notion of validity (e.g., no double spending). The algorithm Π is parameterized by V (denoted by Π^{V}) to ensure that nodes always maintain valid ledgers.

Non-flat mining model. All nodes interact with a random oracle $H : \{0,1\}^* \to \{0,1\}^\kappa$ (where κ is the security parameter) that takes an arbitrary length string and output a random value in $\{0,1\}^\kappa$. In each round, a node $i \in \mathsf{N}$ can make at most q_i queries to the random oracle H for some non-negative integer q_i. The integer q_i represents the node's "mining power". We denote $Q = \sum_{i \in \mathsf{N}} q_i$ as the total mining power of all nodes in N. The total mining power of all malicious nodes is at most $\rho \cdot Q$ where $\rho \in (0, 1/2)$ is the fraction of adversarial hash power. This generalizes the "flat" model in [88] in which all nodes have the same mining power.

The non-flat mining model allows nodes with zero-mining power in the SNM model. The adversary can control a large number of (sybil) nodes with zero mining power, making the fraction of malicious nodes η can be arbitrarily close to 1.

Point-to-point communication. Any two nodes can establish a new connection or drop an existing connection. A node can only send messages to its neighbors, to whom it has established connections. These messages are guaranteed to be delivered to the neighbors after some *unknown delivery delay* $\delta \in \mathbb{N}$ rounds.

Establishing/dropping connections. It takes $t_{\mathrm{e}} \in \mathbb{N}$ rounds to establish a new connection between two nodes. At round r, two nodes start establishing a connection with each other. Then, at round $r + t_{\mathrm{e}}$, they can start exchanging messages. Once the connection is established, the nodes can instantly drop it.

Delivering messages. A node u only sends messages to a node v if there is a connection between u and v in that round. The adversary $\mathcal{A}$ is responsible for delivering all messages. The adversary cannot forge or alter the messages of honest nodes, but it can delay or reorder the messages as long as they are delivered within some *unknown delivery delay* $\delta \in \mathbb{N}$ rounds.

Connection observation. For the static adversary, we consider that newly established connections can be instantly observed by the adversary. In Section 2.4, we discuss the adaptive adversary and restrict their ability to observe connections. Specifically, the adversary can only observe new connections after a certain amount of time has passed.

A blockchain execution. The execution of the blockchain protocol is directed by the environment $\mathcal{Z}$, which initiates the set of nodes N. The nodes are initially designed as honest or malicious. We consider the execution with a polynomial in κ number of rounds. In each round, each node receives a list of transactions txs as input from the environment and adjusts its connections according to Π. Each honest node executes Π and incorporates any input and messages from its neighbors into its local data structure $\mathcal{C}$. Each node $i \in \mathsf{N}$ accesses the random oracle $H(\cdot)$ exactly q_i times with different nonces to find a valid proof of work (PoW). If an oracle call

returns a proof of work, the node can generate a new block and send the new block to its neighbors, who will propagate it further.

Static adversary. We start with a static adversary $\mathcal{A}$ that controls the same subset of nodes throughout the entire execution of the protocol. In Section 2.4, we will analyze the security of our protocols against *(weakly) adaptive adversaries* that can corrupt more nodes during the protocol execution, but in certain restricted manners. To enable security in the presence of weakly adaptive adversaries, we will consider a restricted point-to-point communication model in which the adversary can only observe the connections among honest nodes after a sufficiently long time.

For adversary $\mathcal{A}$, and environment $\mathcal{Z}$, let $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ be a random variable denoting the joint view of all nodes (i.e., all their inputs, randomness, and messages received) over all rounds.

We will be considering executions with restricted adversaries and environments. Given a predicate $\Gamma_{\text{stat}}(\cdot,\cdot,\cdot,\cdot,\cdot)$, we define Γ_{stat}-admissible environment as follows.

Definition 1 (Γ_{stat}-admissible environments)**.** *We say a tuple* $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ *with* $\Gamma_{\text{stat}}(n, Q, \eta, \rho, \delta) = 1$ *is* Γ_{stat}*-admissible w.r.t* $(\Pi^{\mathsf{V}}, \Lambda)$ *if* $\mathcal{A}$*,* $\mathcal{Z}$ *are probabilistic polynomial-time algorithms, and for every* VIEW *in the support of* $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$*, the following holds:*

1. *The adversary* $\mathcal{A}$ *can control at most* $\eta \cdot n$ *nodes, where* n *is the total number of nodes. The total mining power of the nodes controlled by the adversary is at most* ρQ*, where* Q *is the total mining power.*
2. *In every round* r*, the environment* $\mathcal{Z}$ *only sends the list of transactions* txs *as input to an honest player that maintains a ledger* $\mathcal{L} = \Lambda(\mathcal{C})$ *if the list of transactions can be appended to the ledger, i.e.,* $\mathsf{V}(\mathcal{L}\|txs) = 1$*.*
3. *A message sent by an honest node is delayed by at most* δ *rounds before sending*

to its neighbors.

4. *The adversary can instantly observe newly established connections among nodes.*

When context is clear, we refer $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ as Γ_{stat}-admissible.

2.1.2 Security Properties

We say a blockchain protocol is secure if it maintains a ledger that satisfies the *persistence* and *liveness* properties. The persistence property states that if a transaction is added to the ledger of an honest player, then it will be added to the same position in the ledgers of all honest players. The liveness property states that if a transaction is given as input to all honest players, the transaction should eventually appear on the ledgers of honest players. The security analysis [88] of a blockchain protocol relies on a property that any valid message can be disseminated from any honest node within a bounded time. We refer to this property as *reliable dissemination*.

Notations. Let $\mathcal{C}_i^r$ be the local chain of player i at round r. We define the persistence and liveness properties in the joint view $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ as follows. We define $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ to denote that VIEW in the support of $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$. We define a function $\varepsilon(\cdot)$ is a *negligible* such that for any polynomial $\mathsf{poly}(\cdot)$, there exists some $\kappa_0 \in \mathbb{N}$ in which $\forall \kappa \geq \kappa_0, \varepsilon(\kappa) \leq \frac{1}{\mathsf{poly}(\kappa)}$.

Persistence. For a $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{per}(\text{VIEW}) = 1$ iff for any honest player i at round r and any transaction tx in the ledger $\Lambda(\mathcal{C}_i^r)$, and any honest player j at round $r' \geq r$, we have $\Lambda(\mathcal{C}_i^r, tx) = \Lambda(\mathcal{C}_j^{r'}, tx)$.

Definition 2 (Persistence)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* persistence in Γ_{stat}-environments *if* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{per}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa)$.

Liveness. For $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ we define $\mathsf{liv}(\text{VIEW}) = 1$ iff there exists a

"wait time" parameter $\mathsf{t} = O(\kappa)$ such that for any transaction tx that is given as input for all the honest players from round r to round $r + \mathsf{t}$, we have, any honest player i at round $r + \mathsf{t}$, $\Lambda(tx, \mathcal{C}_i^{r+\mathsf{t}}) > 0$.

Definition 3 (Liveness)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* liveness in Γ_{stat}-environments *if* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{liv}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa)$.

Reliable dissemination. The *reliable dissemination* property states that upon an honest node receives (or generates) a valid message, with high probability, the message will be disseminated to all (honest) nodes in the network within a bounded time. For a $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, consider a message msg that is first known by an honest player i at round r_{msg}. Here, the message msg could either be generated by the honest player i or sent from the adversary. For a parameter $\Delta \in \mathbb{N}$, we define $\mathsf{rel}(\text{VIEW}, \Delta) = 1$ iff for any message msg that is known by an honest node at round r_{msg}, all honest nodes received the message msg at round $r_{msg} + \Delta$.

Definition 4 (Reliable dissemination)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* Δ-reliable dissemination in Γ_{stat}-environments *if*

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{rel}(\text{VIEW}, \Delta) = 1] \geq 1 - \varepsilon(\kappa).$$

The security analysis in Pass et al. [88] uses reliable dissemination[2] as an assumption. While this condition holds for fully connected networks, it may not necessarily hold for real-world blockchain networks, e.g., Bitcoin.

[2]To be precise, the security analysis in Pass et al. [88] assume a *perfect reliable dissemination*, i.e., the messages from honest nodes will *always* be disseminated to all other honest nodes.

2.1.3 Consensus in Sparse Network

We now introduce the notion of network sparsity and then define the "consensus in sparse network" problem.

Network sparsity. The *sparsity* of a network is measured as *the average number of connections per honest node within a given period.* Consider a blockchain protocol execution, a VIEW in the support of $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, and a parameter $t_{\mathrm{s}} = \Omega(\kappa)$. For a round r, let m_r be the number of connections that are established at some round $r' \in [r, r + t_{\mathrm{s}}]$ and have at least one end is an honest node. Let n_r be the number of nodes that are honest at round $r + t_{\mathrm{s}}$. We define the sparsity as $d_r = \frac{m_r}{n_r}$, the average number of connections per honest node from round r to round $r + t_{\mathrm{s}}$. For parameters d, t_{s}, we define $\mathsf{spa}(\text{VIEW}, d, t_{\mathrm{s}}) = 1$ iff for any round r in the execution VIEW, we have, $d_r \leq d$.

Definition 5 (Network sparsity)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* d-sparsity in Γ_{stat}-environments *if there exists a parameter* $t_{\mathrm{s}} = \Omega(\kappa)$ *such that*

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{spa}(\text{VIEW}, d, t_{\mathrm{s}}) = 1] \geq 1 - \varepsilon(\kappa).$$

We define the consensus in sparse network as follows.

Definition 6 (Consensus in Sparse Network)**.** *The goal is to construct a consensus protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *in a* Γ_{stat}*-admissible environments that can achieve together 1)* persistence *and* liveness*; and 2)* sparsity*.*

2.2 Protocol Design

The CoSpaN protocol extends Nakamoto's protocol [82] and implements a P2P network protocol that provides reliable dissemination. In CoSpaN protocol, nodes form a new network topology every epoch, which is a duration of L consecutive

blocks. During an epoch, nodes use merge-mining to generate readily available proof-of-mining power and establish pseudo-identities called *core registrations*, which include public keys and a commitment to their network addresses. Each registration entitles a node to act as a *core node* in the next epoch. Thus, a node with high mining power is often associated with multiple (logical) core nodes. Nodes that are not associated with any core nodes are referred to as *periphery nodes*. Based on the selection of core nodes, CoSpaN protocol constructs a *core-periphery topology* that connects all honest nodes in a small-diameter network.

2.2.1 Protocol design

Protocol Π_{CSN} employs an *epoch-based reconfiguration*, where a new topology is constructed every epoch. The reconfiguration in each epoch is divided into two stages: *core selection* and *topology construction*. The pseudocode of protocol Π_{CSN} can be found in Figure 4.

2.2.1.1 Epoch-based configuration

As shown in Figure 5, each epoch consists of a consecutive group of L (where $L \in \mathbb{N}$) blocks. For a node in the network, the i-th epoch will include all rounds in which the block heights are in the interval $[(i-1)L, iL)$. The block height at a round is defined as length of the longest chain. Due to network delays and malicious behavior, nodes may observe different block heights.

Each epoch is divided into two smaller periods: a *registration period* for core selection, followed by a short *grace period* for topology construction. For $k = \Omega(\kappa)$ and $k \ll L$, we set the lengths of the registration and grace periods to $L_{\texttt{reg}} = L - 2k$ and $2k$, respectively. The grace period serves to tolerate any potentially different views of the nodes. During the first half of the grace period, consisting of k blocks,

CoSpaN Protocol Π_{CSN}

Parameters:

- Epoch length L, grace period half-length $k = \Omega(\kappa)$
- Core width s, connection parameter d

Local state: Node u has a state $\langle \mathcal{C}, \mathsf{ip} \rangle$ containing blockchain copy $\mathcal{C}$ and its network address ip.

Compute block height $l = len(\mathcal{C})$
Compute epoch number $i = \lfloor l/L \rfloor + 1$
Core selection: If $l < L - 2k$
  Generate a pseudo-identity cid
  Set $\mathsf{context} := \langle \mathsf{prev}, MkRoot(txs), \mathsf{cid} \rangle$
  Upon finding a *nonce* such that
    $H(\mathsf{context}, nonce) \in [\mathsf{T}, \mathsf{T} + \mathsf{T}_{\text{core}}]$ (Eq. 2.2),
  propagates a core registration
    $\mathsf{cr} = \langle \mathsf{context}, nonce \rangle$.

Topology construction: If $l \in [iL - k, iL)$

Randomness extraction:
If $l = iL - k$, concatenate all the blocks with block heights in $[iL - 3k, iL - 2k)$ and return the hash value as rnd_i

Verifiable random connections:
Let $\bar{\mathsf{C}} = \{$core registrations in the registration period$\}$
Let $\bar{\mathsf{C}}_u \subseteq \bar{\mathsf{C}}$ be the set of core registrations from node u
Core-core connections:
For each core registration $\mathsf{cr} \in \bar{\mathsf{C}}_u$ **do**

- *Step 1: Announcing eligible connections*
  For each core registration $v \in \bar{\mathsf{C}} \setminus \bar{\mathsf{C}}_u$ **do**
    With probability $\bar{\mu}$, announce the address to v via a eligible message.
- *Step 2: Establishing connections*
  For each eligible message from w with the address $w.\mathsf{ip}$
    Connect to $w.\mathsf{ip}$ after verification.
- *Step 3: Accepting connections*
  For each connection request
    Accept the connection after verification.

Core-periphery connections:
[u as a core node] Establishing connections
For each core registration $\mathsf{cr} \in \bar{\mathsf{C}}_u$ **do**
  For each periphery node $v \neq u$ with address $v.\mathsf{ip}$ **do**
    With probability $\bar{\mu}$, connect to $v.\mathsf{ip}$.
[u as a periphery node] Accepting connections
For each connection request from w
  Accept the connection after verification.

Figure 4: CoSpaN protocol Π_{CSN}.

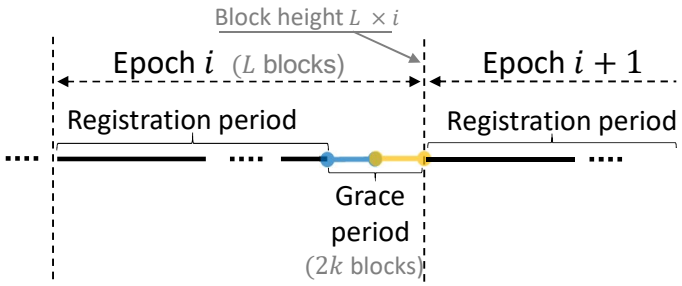


Figure 5: Epoch of L blocks consisting of 1) a registration period in which nodes use a PoW merged-mining to perform core registration and 2) a short grace period of $2k$ blocks for $k = O(\kappa)$. The first half of the grace period is to guarantee the convergence of nodes' views on the set of core registrations. In the second half of the grace period, nodes will construct a new network topology using verifiable random connections, computed from the core registrations.

there is sufficient time to rule out any divergence among nodes regarding the set of core registrations, i.e., the set of core nodes in the next epoch. The k blocks in the second half allow sufficient time for nodes to establish new connections. During the grace period, the nodes maintain their existing connections and only drop them when the new epoch begins to ensure network connectivity continuity.

2.2.1.2 Core selection

During the registration period, the nodes are selected via the same PoW mining to find new blocks, albeit, with different thresholds [92, 8]. To generate new blocks, nodes attempt to solve a PoW puzzle by searching for a *nonce* over a context that satisfies the hash inequality

$$H(\mathsf{context}, nonce) < \mathtt{T},$$

where $\mathtt{T}$ is the mining difficulty. The context consists of a hash value prev of the previous block, the Merkle root $MkRoot(txs)$, a single hash value to validate the included transactions, and a network pseudo-identity cid, i.e.,

$$\mathsf{context} = \langle \mathsf{prev}, MkRoot(txs), \mathsf{cid} \rangle. \tag{2.1}$$

Let $\mathtt{T}_{\text{core}}$ denotes *core registration difficulty*, whenever the node finds a *nonce* that

$$\mathtt{T} \leq H(\mathsf{context}, nonce) < \mathtt{T} + \mathtt{T}_{\text{core}}, \tag{2.2}$$

it generates and propagates a core registration

$$\mathsf{cr} = \langle \mathsf{context}, nonce \rangle.$$

The core registrations will be propagated within the P2P network and included in the ledger as (prioritized) transactions.

Denote by s the *core width*, which represents the expected number of core registrations (i.e. the number of core nodes) in each epoch. To ensure $\Theta(s)$ core registrations in each period, we set $\mathtt{T}_{\text{core}} = \frac{s}{L_{\mathtt{reg}}}\mathtt{T}$. This can be verified by noting that the mining difficulty T leads to approximately $L_{\mathtt{reg}}$ blocks during the registration period. Our later security analysis remains valid when the number of actual registrations is within a constant ω, such as $\omega = 90\%$, of the core width s. This adds robustness to the protocol, for example against low participation of nodes in the core selection.

2.2.1.3 Core-periphery topology construction

During the second half of the grace period, the *core-periphery topology* is constructed in two phases. First, all nodes in the network *extract randomness* from the previous epoch. Then, based on this randomness, core nodes establish *verifiable ran-*

dom connections to other core and periphery nodes. The probability that a core node establishes a connection to another (core or periphery) node is $\bar{\mu} = \frac{d}{2s}$, where d is the *connection parameter* and s is the *core width*. Note that the expected number of (logical) core nodes associated with the same (physical) node (or miner) i is proportional to its mining power q_i, as is the expected number of connections established by i.

Randomness extraction. Similar to the method described in [34], nNodes compute the randomness rnd_i as the hash value of the concatenation of the last k blocks in the registration period, which have block heights in the range $[iL - 3k, iL - 2k)$. This ensures that all nodes obtain the same randomness rnd_i at the second half of the grace period.

Connection establishment. During the second half of the grace period, all nodes utilize core registrations and extracted randomness to establish new connections. The connection establishment procedure is divided into two sub-procedures for two types of connections: *core-core connections* and *core-periphery connections*.

- Core-core connections. Each core node u establishes a connection with every other core node v with a probability of $\bar{\mu}$. Two core nodes u and v are connected if either u establishes a connection to v or v establishes a connection to u.

- Core-periphery connections. Each core node u establishes a connection with every periphery node v with a probability $\bar{\mu}$. Periphery nodes do not establish connections with other nodes.

2.2.2 Security components

We present the secure components of the CoSpaN protocol. First, we describe how nodes establish verifiable random connections. This ensures that the adversary

cannot monopolize the connections and protect the confidentiality of connections, i.e., each connection is only known by the two participating nodes. Next, we describe how to preserve the confidentiality of core nodes to prevent the adversary from DoS attacking highly connected nodes that associate with multiple core nodes.

2.2.2.1 Verifiable random connections

To establish verifiable random connections, a public-key pseudorandom function known as a VRF is employed [40]. The VRF provides proofs that its outputs were calculated correctly and randomly, making them difficult to predict. The owner of the secret key can compute the function value σ and an associated proof π for a given input value. Others can use an algorithm called Verify, along with the proof and the associated public key, to check if the function value was calculated correctly without revealing any information about the secret key. For the formal definition of the VRF, please refer to Section 2.7.2.

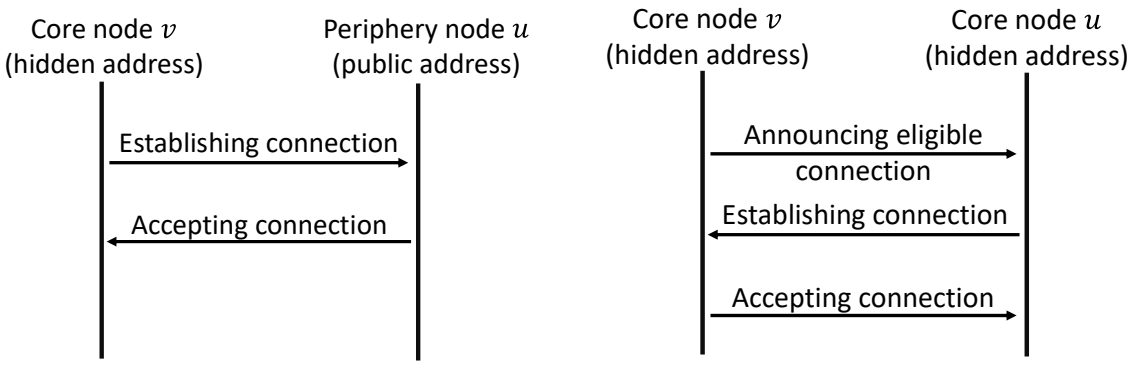
Consider a core node u that hold the key pair $(\mathsf{sk}', \mathsf{pk}')$ for VRF. For a (core or periphery) node v, let id_v be the identity of node v. Here, if v is a core node, we set the identity id_v as the pseudo-identity cid_v of v. If v is a periphery node, we set the identity id_v as the address ip_v of v. The node u computes $(\sigma_v, \pi_v) := \mathsf{Prove}_{\mathsf{sk}'}(\mathsf{rnd}_i || \mathsf{id}_v)$. We say that u is *eligible* to connect to v, in epoch $i+1$, if and only if output σ_v is smaller than a threshold $\mathsf{T}_{\mathrm{con}} = \bar{\mu} \cdot 2^{\kappa}$, i.e.,

$$\sigma_v < \mathsf{T}_{\mathrm{con}}. \tag{2.3}$$

Upon receiving a connection request from u, the node v can verify node u is eligible by verifying 1) (σ_v, π_v) is computed correctly, and 2) $\sigma_v < \mathsf{T}_{\mathrm{con}}$.

2.2.2.2 Confidentiality of core nodes

As we mentioned, we preserve the confidentiality of core nodes by hiding the association between core nodes and their physical addresses. Thus, the node's address is not included in the pseudo-identity to preserve anonymity. To generate cid, a node generates key pairs $(\mathsf{sk}', \mathsf{pk}')$, $(\mathsf{sk}, \mathsf{pk})$, $(\bar{\mathsf{sk}}, \bar{\mathsf{pk}})$ for a VRF, a public-key encryption scheme, and a signature scheme, respectively. The network pseudo-identity is returned as $\mathsf{cid} = \langle \mathsf{pk}', \mathsf{pk}, \bar{\mathsf{pk}} \rangle$.



(a) Core-periphery connection. (b) Core-core connection.

Figure 6: Establishing connections. To preserve core anonymity, an extra step is added to establish core-core connections.

In Figure 6, we show the interactions to establish core-periphery and core-core connections. The establishment of a core-periphery connection consists of two steps. First, a core node u requests to connect to a periphery node v if it is eligible to do so. Then, node v will accept the connection after verifying that u is eligible.

For the core-core connections, as the addresses of core nodes cannot be obtained from the pseudo-identities, an extra step is added to establish core-core connections. First, each core node u need to broadcast an encryption of it address to the core nodes that it is eligible to connect. Node u creates a message M containing infor-

mation on eligible connections to broadcast to all nodes in the P2P network. For each "eligible" core node v, u uses v's public key, $v.\mathsf{pk}$, for encryption. It computes $\xi \leftarrow \mathsf{Enc}_{v.\mathsf{pk}}(\sigma_{\text{core}}, \pi_{\text{core}}, \mathsf{ip})$ and adds ξ to a list M. Node u then computes a signature $\gamma = \textsc{Sign}_{\bar{\mathsf{sk}}}(M, \mathsf{cr})$ and broadcasts (M, cr, γ) to all nodes in the network. By using encryption, u limits the visibility of its network address, $u.\mathsf{ip}$, to only those nodes that u is eligible to connect to in the future. To prevent spamming attacks, every node will only forward at most one valid message (M, cr, γ) for each core registration cr. Then, the core node v can decrypt the message to extract the address of the node u. After verifying u eligible, node v request to make connection to node u, by sending a signature $\textsc{Sign}_{\bar{\mathsf{sk}}}(v.\mathsf{cr})$. The node u then accept the connection if the signature of node v is correct.

2.2.3 Complexity

Computation cost. On average, each core node performs 1) $n+s$ times to compute the VRF (each for a core or periphery node); 2) d times to encrypt (each for an eligible core node); and 3) ds times to decrypt (for every encryption from core nodes). In total, the time complexity of a core node in each epoch is $O(n+d\cdot s)$. On average, each periphery node performs d times to verify the connection. Thus, the time complexity of a periphery node in each epoch is $O(d)$.

Communication cost. In an epoch, each core node broadcasts a message that contains the encryption to all other node. The message complexity to broadcast the encryption is $O(n \cdot s)$. The remaining communication happens between the nodes that are connected to each other in that epoch. The message complexity to this communication is $O(n+s)d$. Thus, the total message complexity in each epoch is $O(n\cdot s+n\cdot d+s\cdot d)$.

2.2.4 Discussions

We now discuss the practicality of CoSpaN protocol in real-world blockchain systems.

CoSpaN as a backbone network. We can use the CoSpaN protocol to construct a secure backbone network for propagating important information, such as block headers. For actual data propagation, a high-performance relay network such as FIBRE [28] or Falcon [10] can be used.

Coping with low participation rate of core nodes. In practice, some core nodes may not participate in establishing connections in the next epoch. This can be due to nodes being offline or behind a NAT, making them unreachable by other nodes. To address this issue without increasing the number of connections, we can prioritize the use of VRF outputs to establish or accept connections instead of relying on the inequalities. Specifically, nodes will establish or accept connections with the top few nodes that have the smallest VRF outputs. For example, suppose a node needs to connect with some core nodes but they did not participate in establishing connections. In that case, the node will instead connect with the core nodes that have the next smallest VRF outputs.

Public-private nodes. In Bitcoin network, nodes can be categorized into two types: public and private nodes. We can also view the Bitcoin network as a core-periphery network where the public nodes can be viewed as core nodes and the private nodes can be viewed as periphery nodes. This is contradict with the CoSpaN network where the core nodes are private and the periphery nodes are public.

The CoSpaN network is more resistant to DoS attacks, compared with the Bitcoin network. Indeed, in Bitcoin network, the IP addresses of core nodes are public (known by everyone); while the IP addresses of core nodes in CoSpaN network are private

(only known by the nodes that have direct connections to them). Without knowing the IP address of core nodes, the adversary cannot perform DOS attacks to bring down those core nodes.

2.3 Security analysis

Threat model	Adaptive corruption	Connection observability	Epoch length L	Core width s	Connection parameter d
Static	n/a	Instantly	$\Omega\left(\frac{\kappa}{(1-\rho)\omega}\right)$	$\Omega\left(\frac{\kappa}{1-\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-\rho)\omega)}+\log s\right)$
M-adaptive	τ_{corr} rounds	Instantly	$O\left(\tau_{\text{corr}}\cdot\alpha\right)$	$\Omega\left(\frac{\kappa}{1-\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-\rho)\omega)}+\log s\right)$
S-adaptive	Instantly	τ_{obs} rounds	$O\left(\tau_{\text{obs}}\cdot\alpha\right)$	$\Omega\left(\frac{\kappa+h}{1-2\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-2\rho)\omega)}+\log s\right)$
S-adaptive$\langle\phi,\gamma\rangle$	Instantly	τ_{obs} rounds	$O\left(\tau_{\text{obs}}\cdot\alpha\right)$	$\Omega\left(\frac{\kappa+h\cdot\phi}{\gamma-\rho}\right)$	$O\left(\frac{\kappa}{-\log((\gamma-\rho)\omega)}+\log s\right)$

Table 2.: Feasible parameter settings of CoSpaN (L, s, d) against different adversary models. CoSpaN, parameterized by the epoch length L, the core width s, and the connection parameter d can achieve reliable dissemination with a sparsity of $\frac{2}{1-\eta}(1+\frac{s}{n})d = O(\log n)$, where n is the number of nodes and η denotes the fraction of malicious nodes. Defending against (weakly) adaptive adversaries requires (1) shorter epoch lengths and (2) larger numbers of core nodes to limit the effect of targeted corruption toward the core nodes. In an S-adaptive$\langle\phi,\gamma\rangle$ model, where $\phi \in (0,1)$ and $\gamma \in (\rho, 1-\rho))$, we consider an S-adaptive adversary when the top ϕ fraction of honest nodes control a fraction γ of mining power.

We prove that by choosing sufficient parameters (Theorem 2.3.3), the CoSpaN protocol can achieve security properties (defined in Section 2.1.2) under the presence of a static adversary in a network with $O(\log n)$ sparsity. Later, in Section 2.4, we will show that the CoSpaN protocol can also achieve security properties under the presence of weakly adaptive adversaries. In Table 2, we provide a summary of the security analysis of the CoSpaN protocol against a static adversary and two additional weakly adaptive adversaries. An M-adaptive adversary needs to *wait for* τ_{corr} *rounds*

to corrupt an honest node. An S-adaptive adversary can corrupt a node instantly, however, needs to *wait for τ_{obs} rounds to discover a newly established connection.* We also consider an S-adaptive$\langle\phi, \gamma\rangle$ setting, in which the top ϕ fraction of honest nodes control a fraction γ of mining power. The S-adaptive setting can be seen as a special case of S-adaptive$\langle\phi, \gamma\rangle$ with $\phi = 1$ and $\gamma = 1 - \rho$.

2.3.1 Security properties

We prove that protocol Π_{CSN} achieves security properties (including persistence and liveness) through *induction* (see Figure 7). In each induction step, we assume reliable dissemination in the current epoch and prove that protocol Π_{CSN} achieves security properties that lead to reliable dissemination in the next epoch. For the basis step, we assume a *bootstrapping condition* that provides reliable dissemination for protocol Π_{CSN} in the first epoch.

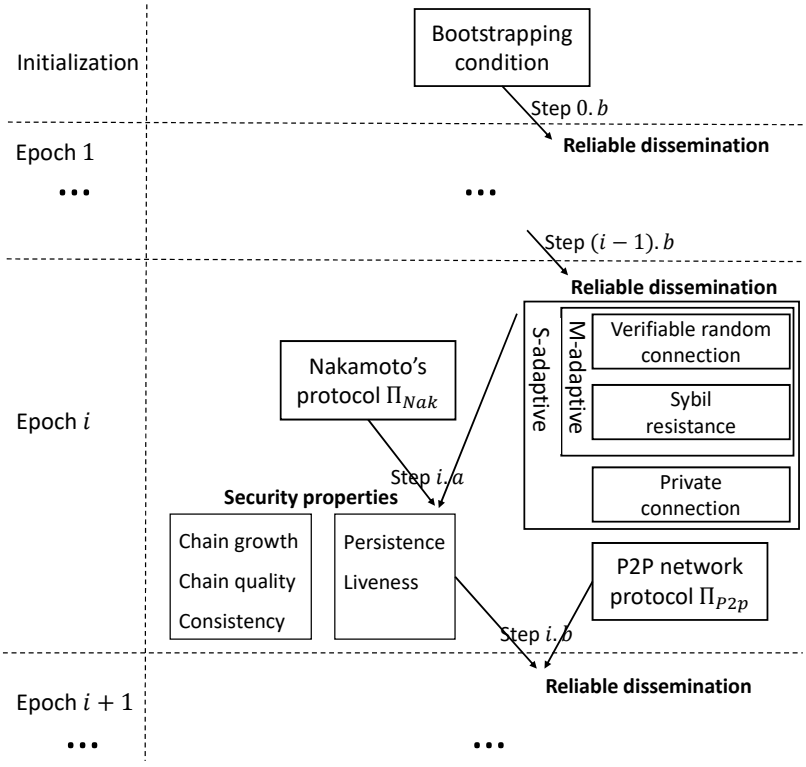


Figure 7: The induction proof of the CoSpaN protocol.

We prove that the Π_{CSN} protocol achieves the necessary security properties for the ledger (persistence and liveness) and consensus (chain growth, chain quality, and consistency). For simplicity, we will refer to both the security properties for the ledger and consensus as the security properties for the remainder of this paper. For a given $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\text{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{sec}(\text{VIEW}) = 1$ if the following conditions are met.

- *Persistence and liveness.* $\mathsf{per}(\text{VIEW}) = 1$ and $\mathsf{liv}(\text{VIEW}, \mathsf{t}) = 1$ (see Subsection 2.1.2).

- *Chain growth.* There exists a parameter $\mathsf{cg} > 0$ such that, for any honest player i at any rounds r and $r' = r + \Omega(\kappa)$, we have, $\mathsf{len}(\mathcal{C}_i^{r'}) - \mathsf{len}(\mathcal{C}_i^{r}) \geq (r' - r)\mathsf{cg}$, where, $\mathcal{C}_i^r, \mathcal{C}_i^{r'}$ are the chains of the player i at round r, r', respectively.

- *Chain quality.* There exists a parameter $\mathsf{cq} > 0$ such that, for any honest player i at any rounds r, among any $k = \Omega(\kappa)$ consecutive blocks in the chain $\mathcal{C}_i^r$, the fraction of honest blocks is at least cq.

- *Consistency.* For any honest player i at rounds r and any honest player j at rounds $r' > r$, after removing the last k block of $\mathcal{C}_i^r$, we obtain a prefix of the chain $\mathcal{C}_j^{r'}$.

Let $e_{\max}$ be the number of epochs in the execution. For $i \in [0..e_{\max}]$, let VIEW_i be the view of nodes in VIEW up until the last round of epoch i (the last round in which there exists an honest node with the chain of length $i \cdot L$). For $\Delta = O(\log n)\delta$, we define $\mathcal{S}_i, \mathcal{R}_i$ as the events where protocol Π_{CSN} achieves the *security properties* and *reliable dissemination*, respectively, up until epoch i:

$$\mathcal{S}_i \stackrel{\text{def}}{=} (\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}_i) = 1),$$
$$\mathcal{R}_i \stackrel{\text{def}}{=} (\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{rel}(\text{VIEW}_i, \Delta) = 1).$$

In the induction proof, for each epoch $j \in [1..e_{\max}]$, we prove

$$\Pr[\mathcal{S}_j \wedge \mathcal{R}_{j+1}] > 1 - (2j+1)\varepsilon(\kappa).$$

The statement for $j = e_{\max}$ indicates that protocol Π_{CSN} achieves the security properties.

Basis step (step 0-B). We assume a *bootstrapping condition* where the network of honest nodes is connected with a small diameter at epoch 1. Thus, protocol Π_{CSN} achieves reliable dissemination at epoch 1, i.e., $\Pr[\mathcal{R}_1] > 1 - \varepsilon(\kappa)$. The bootstrapping condition can be achieved through a trusted setup or a one-time decentralized bootstrapping scheme.

Induction step. Given that the hypothesis is true for $j = i - 1$, (where $i \in [1..e_{\max}]$), we will prove the statement is true of $j = i$. The induction step consists of two parts as follows.

- *Step i-A:* Given from step $(i-1)$-B that

  $$\Pr[\mathcal{S}_{i-1} \wedge \mathcal{R}_i] > 1 - (2i-1)\varepsilon(\kappa).$$

  We prove $\Pr[\mathcal{S}_i \wedge \mathcal{R}_i] > 1 - 2i\varepsilon(\kappa)$ by showing that

  $$\Pr[\mathcal{S}_i \mid \mathcal{S}_{i-1} \wedge \mathcal{R}_i] > 1 - \epsilon(\kappa).$$

- *Step i-B:* Given from step i-A that

  $$\Pr[\mathcal{S}_i \wedge \mathcal{R}_i] > 1 - 2i\varepsilon(\kappa).$$

  We prove $\Pr[\mathcal{S}_i \wedge \mathcal{R}_{i+1}] > 1 - (2i+1)\varepsilon(\kappa)$ by showing that

  $$\Pr[\mathcal{R}_{i+1} \mid \mathcal{S}_i \wedge \mathcal{R}_i] > 1 - \epsilon(\kappa).$$

2.3.1.1 Step i-A: Achieving the security properties from reliable dissemination

Assuming that Π_{CSN} has achieved the security properties up until epoch $i-1$, we will now prove that it also achieves security properties up until epoch i. First, we summarize the security analysis of Nakamoto's protocol in the PSS model [88]. Then, we can reduce the security of Π_{CSN} in the SNM model to the security of Nakamoto's protocol Π_{Nak} in the PSS model, provided that the CoSpaN protocol achieves reliable dissemination.

Nakamoto's protocol in PSS model [88]. We summarize the security analysis of Nakamoto's protocol Π_{Nak} in PSS model [88], where nodes can send messages to all other nodes within a bounded time. We then demonstrate that the security of the CoSpaN protocol Π_{CSN} can be reduced to the security of Π_{Nak} if the reliable dissemination property is achieved. We say a tuple $(n, \rho, \Delta, \mathcal{A}, \mathcal{Z})$ with $\Gamma_{\text{PSS}}(n, \rho, \Delta) = 1$ is Γ_{PSS}-admissible w.r.t (Π, Λ) if reliable dissemination is given as an assumption, i.e., any message sent by an honest node is delayed by at most Δ rounds before sending to all other nodes. Further, Γ_{PSS}-admissible environments, in each round, each node can make 1 query to the random oracle (flat model).

Let $p = \frac{\mathsf{T}}{2^\kappa}$ be the probability that a node can generate a new block with a single query. We consider the following two quantities. Let $\alpha(n, \rho, \Delta) = 1 - (1-p)^{(1-\rho)n}$ be the probability that at least one honest node generates a new block in one round. Let $\beta(n, \rho, \Delta) = \rho n p$ be the expected number of blocks that the adversary can mine in a round. When the n, ρ, Δ is clear from the context, we simply write α, β. The analysis in [88] considers environments with a predicate Γ^*_{PSS} with the *honest majority assumption*. For $n, \Delta \in \mathbb{N}$, $\rho \in (0,1)$, we say $\Gamma^*_{\text{PSS}}(n, \rho, \Delta) = 1$ if $\alpha(1-2(\Delta+1)\alpha) > \beta$.

Definition 7 (Γ_{PSS}-admissible environments)**.** *We say a tuple* $(n, \rho, \Delta, \mathcal{A}, \mathcal{Z})$ *with*

$\Gamma_{\text{PSS}}(n, \rho, \Delta) = 1$ *is* Γ_{PSS}*-admissible w.r.t* (Π, Λ) *if* $\mathcal{A}$*,* $\mathcal{Z}$ *are probabilistic polynomial-time algorithms, and for every* $\kappa \in \mathbb{N}$*, for every view in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa)$*, the following holds:*

1. *The adversary* $\mathcal{A}$ *can control at most* $\rho \cdot n$ *nodes, where* n *is the total number of nodes, and each node has* 1 *mining power.*
2. *Conditions (2) in Definition 1.*
3. *Any message sent by an honest node is delayed by at most* Δ *rounds before sending to all other nodes.*

We can achieve the security properties for ledger as follows.

Lemma 8 (Theorem 7.4 in [88])**.** *Consider protocol* Π_{Nak} *in* Γ^*_{PSS}*-admissible environments. Protocol* Π_{Nak} *maintains a ledger that satisfies persistence and liveness with the "wait time"* $\mathsf{t} = \frac{\kappa}{(1-\epsilon)\alpha}$*.*

In Pass et al. [88], the security properties for ledger are proven based on three *security properties for consensus* called *chain growth*, *common prefix*, and *chain quality*.

Chain growth. For a chain growth parameter cg and $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{grw}(\text{VIEW}, \mathsf{cg}) = 1$ iff for any honest player i at any rounds r and $r' = r + t$ (where $t = \Omega(\kappa)$), we have, $\mathsf{len}(\mathcal{C}_i^{r'}) - \mathsf{len}(\mathcal{C}_i^{r}) \geq t\mathsf{cg}$ where, $\mathcal{C}_i^r, \mathcal{C}_i^{r'}$ are the chains of the player i at round r, r', respectively.

Definition 9 (Chain growth)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* chain growth in Γ_{stat}-environments *if there exists a chain growth parameter* $\mathsf{cg} > 0$ *such that*

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{grw}(\text{VIEW}, \mathsf{cg}) = 1] \geq 1 - \varepsilon(\kappa),$$

in all Γ_{stat}*-admissible environments.*

Chain quality. For a chain quality parameter cq and $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{qty}(\text{VIEW}, \mathsf{cq}) = 1$ iff for any honest player i at any rounds r, among any $k = \Omega(\kappa)$ consecutive blocks in the chain $\mathcal{C}_i^r$, the fraction of honest blocks is at least cq. and $r' = r + t$, we have, $\mathsf{len}(\mathcal{C}_i^{r'}) - \mathsf{len}(\mathcal{C}_i^r) \geq t \cdot \mathsf{cg}$.

Definition 10 (Chain quality)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* chain quality in Γ_{stat}-environments *if there exists a chain quality parameter* $\mathsf{cq} > 0$ *such that*

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{qty}(\text{VIEW}, \mathsf{cq}) = 1] \geq 1 - \varepsilon(\kappa),$$

in all Γ_{stat}*-admissible environments.*

Common prefix. For a $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{cns}(\text{VIEW}) = 1$ iff for any honest player i at any rounds r and any honest player j at any rounds $r' > r$, after removing the last $k = \Omega(\kappa)$ block of $\mathcal{C}_i^r$, we obtain a prefix of the chain $\mathcal{C}_j^{r'}$.

Definition 11 (Common prefix)**.** *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* chain quality in Γ_{stat}-environments *if*

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{cns}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa),$$

in all Γ_{stat}*-admissible environments.*

The Nakamoto's protocol achieves chain growth, chain quality, and consistency as follows.

Lemma 12 (Theorem 4.1 (chain growth), Theorem 4.3 (consistency), Theorem 4.2 (chain quality) in [88])**.** *Consider Nakamoto's protocol in* Γ^*_{PSS}*-admissible environments. Nakamoto's protocol achieves chain growth with parameter* $\mathsf{cg} = (1 - \epsilon)\gamma$ *where* $\epsilon > 0$, $\gamma = \frac{\alpha}{1+\Delta\alpha}$*; chain quality with parameter* $\mathsf{cq} = 1 - (1 + \epsilon)\frac{\beta}{\gamma}$*; and consistency.*

CoSpaN in SNM model (given reliable dissemination). Given the event $\mathcal{R}_i$, which signifies that protocol Π_{CSN} achieves reliable dissemination up until epoch i, we can reduce the security of Π_{CSN} in the SNM model to that of Nakamoto's protocol Π_{Nak} in the PSS model. Intuitively, since any valid message can be disseminated within a bounded time Δ, we can simulate an execution in the PSS model to achieve the same message delay as in the SNM model. Additionally, in the non-flat mining model, each node v can be considered as a cluster of multiple nodes in the flat mining model. Similar to [88], we consider environments with a predicate Γ^*_{stat} that assumes an honest majority. For $n, Q, \delta \in \mathbb{N}$, $\eta, \rho \in (0,1)$; we say $\Gamma^*_{\text{stat}}(n, Q, \eta, \rho, \delta) = 1$ if there exists some $\Delta \in \mathbb{N}$ such that, $\Delta = O(\log n)\delta$ and $\Gamma^*_{\text{PSS}}(Q, \rho, \Delta) = 1$.

Lemma 13 (Step i-A)**.** *Consider protocol Π_{CSN} in Γ^*_{stat}-admissible environments. For any $i \in \mathbb{N}$, we have,* $\Pr[\mathcal{S}_i \mid \mathcal{S}_{i-1} \wedge \mathcal{R}_i] > 1 - \epsilon(\kappa)$.

Proof of Lemma 13. We will reduce the security of Π_{CSN} in Γ^*_{stat}-admissible environments to the security of Π_{Nak} in Γ^*_{PSS}-admissible environments.

More concretely, consider a tuple $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ that is Γ^*_{stat}-admissible with respect to Π_{CSN}. We will construct an environment $\mathcal{Z}'$ and an adversary $\mathcal{A}'$ such that the tuple $(Q, \rho, \Delta, \mathcal{A}', \mathcal{Z}')$ is Γ^*_{PSS}-admissible with respect to Π_{CSN}. Based on Lemma 8, protocol Π_{Nak} achieves persistence and liveness in Γ^*_{PSS}-admissible environments. Thus, we can prove the security of Π by mapping the execution of protocol Π_{CSN} in $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ to the execution of protocol Π_{Nak} in (Q, ρ, Δ). Indeed, consider the execution $\mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa)$ and a modified execution in which after some round r, nodes run the protocol Π_{CSN} in the presence of $\mathcal{Z}'$ and $\mathcal{A}'$. We will show that by fixing the randomness, the views in the support of the two executions are *compatible* with high probability.

We say VIEW_1 and VIEW_2 are *compatible*, denoted by $\text{VIEW}_1 \overset{\text{c}}{=} \text{VIEW}_2$, if at any

round r, for any honest node i in VIEW_1, there exists an honest node j in VIEW_2 such that the local chain of i and j are the same.

We construct an environment $\mathcal{Z}'$ from the environment $\mathcal{Z}$ as follows:

- For each node i that is activated by $\mathcal{Z}$, if $q_i > 0$, the environment $\mathcal{Z}'$ initiates a set $F(i)$ of q_i nodes in which each node has 1 mining power.
- If the environment $\mathcal{Z}$ corrupts a node i, the environment $\mathcal{Z}'$ corrupts all nodes in $F(i)$.
- In each round, the environment $\mathcal{Z}'$ sends the same input that $\mathcal{Z}$ sends to i to all nodes in $F(i)$.
- In each round, nodes in $F(i)$ receive the same results from the random oracle queries as the node i.

We also construct a new adversary $\mathcal{A}'$ from the adversary $\mathcal{A}$ as follows:

- Upon $\mathcal{A}$ sending a message to an honest node i (from a malicious node), the adversary $\mathcal{A}'$ sends the same message to all nodes in $F(i)$.
- Consider a message m that is generated at an honest i. Let $\Delta_{ij}(m)$ be the time it takes for node i to send a message to node j in the point-to-point communication model. Upon a node in $F(i)$ broadcasting the message m, the adversary $\mathcal{A}'$ immediately sends m to all nodes in $F(i)$ and delays for $\min(\Delta, \Delta_{ij}(m))$ rounds before sending the message to all nodes in $F(j)$.

As $\mathcal{A}$ and $\mathcal{Z}$ are probabilistic polynomial-time algorithms, we have that $\mathcal{A}'$ and $\mathcal{Z}'$ are also probabilistic polynomial-time algorithms. Additionally, $\Gamma^*_{\text{stat}}(n, Q, \eta, \rho, \delta) = 1$, so $\Gamma^*_{\text{PSS}}(Q, \rho, \Delta) = 1$, meaning that $(Q, \rho, \Delta, \mathcal{A}', \mathcal{Z}')$ is Γ^*_{PSS}-admissible with respect to Π_{CSN}.

Let $\mathsf{EXEC1}(\lambda)$ and $\mathsf{EXEC2}(\lambda)$ denote the output of the executions $\mathsf{EXEC}_{\Pi_{\mathrm{CSN}},\mathcal{A},\mathcal{Z}}(\kappa)$ and $\mathsf{EXEC}_{\Pi_{\mathrm{Nak}},\mathcal{A}',\mathcal{Z}'}(\kappa)$, respectively, with the randomness fixed to λ. Here, the randomness λ can be partitioned into two parts: λ_1 is used when nodes query the random oracle to generate new blocks, and λ_2 is used when nodes generate the keys of the VRFs.

Let w_i be the latest round in $\mathsf{EXEC1}(\lambda)$ such that the maximum epoch in the view of an honest player is i. Let $\mathsf{EXEC1}_{w_i}(\lambda)$ and $\mathsf{EXEC2}_{w_i}(\lambda)$ be the joint view of all nodes in $\mathsf{EXEC1}(\lambda)$ and $\mathsf{EXEC2}(\lambda)$, respectively, until round w_i.

We consider the case where $\mathsf{rel}(\mathsf{EXEC1}_{w_i}(\lambda), \Delta) = 1$, i.e., any honest node in $\mathsf{EXEC1}_{w_i}(\lambda)$ can disseminate any message to all honest nodes within Δ rounds. In this case, for any honest node i and any message m, the nodes in $A(i)$ (in $\mathsf{EXEC2}_{w_i}(\lambda)$) receive m (from other honest nodes and the adversary) at the same round as the node i (in $\mathsf{EXEC1}_{w_i}(\lambda)$). Thus, at any round r, the local chain of i is the same as the local chain of all nodes in $A(i)$, i.e., $\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{\mathrm{c}}{=} \mathsf{EXEC2}_{w_i}(\lambda)$.

We have,

$$\Pr_{\lambda}[\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{\mathrm{c}}{=} \mathsf{EXEC2}_{w_i}(\lambda)] \geq \Pr_{\lambda}[\mathsf{rel}(\mathsf{EXEC1}_{w_i}(\lambda), \Delta)]. \tag{2.4}$$

Consider an honest node i at round r, and an honest node j at round $r' \geq r$ in $\mathsf{EXEC1}_{w_i}(\lambda)$. If $\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{\mathrm{c}}{=} \mathsf{EXEC2}_{w_i}(\lambda)$, then there exists an honest node i' at round r and an honest node j' at round r' in $\mathsf{EXEC2}_{w_i}(\lambda)$ such that the local chain of i at round r (respectively, the local chain of j at round r') is the same as the local chain of i' at round r (respectively, the local chain of j' at round r'). Thus, based on Definition 2, if $\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{\mathrm{c}}{=} \mathsf{EXEC2}_{w_i}(\lambda)$, then $\mathsf{per}(\mathsf{EXEC2}_{w_i}(\lambda)) = \mathsf{per}(\mathsf{EXEC1}_{w_i}(\lambda))$. Using similar arguments for liveness and reliable dissemination, we

have,

$$\text{If } \mathsf{EXEC1}_{w_i}(\lambda) \stackrel{c}{=} \mathsf{EXEC2}_{w_i}(\lambda),$$

$$\text{then } \mathsf{sec}(\mathsf{EXEC1}_{w_i}(\lambda)) = \mathsf{sec}(\mathsf{EXEC2}_{w_i}(\lambda)).$$

Hence, we have,

$$\begin{aligned}
&\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}) = 1] \\
=&\Pr_{\lambda}[\mathsf{sec}(\mathsf{EXEC1}(\lambda)) = 1] \\
\geq&\Pr_{\lambda}[\mathsf{sec}(\mathsf{EXEC2}(\lambda)) = 1] \times \Pr_{\lambda}[\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{c}{=} \mathsf{EXEC2}_{w_i}(\lambda)] \\
\geq&(1 - \varepsilon(\kappa)) \times \Pr_{\lambda}[\mathsf{EXEC1}_{w_i}(\lambda) \stackrel{c}{=} \mathsf{EXEC2}_{w_i}(\lambda)] \quad \text{(Lemma 8)} \\
\geq&\Pr_{\lambda}[\mathsf{rel}(\mathsf{EXEC1}_{w_i}(\lambda), \Delta)] - \varepsilon(\kappa) \quad \text{(Eq. 2.4)}.
\end{aligned}$$

□

2.3.1.2 Step i-B: Achieving reliable dissemination from the security properties

Given that protocol Π_{CSN} achieves the security properties *up until epoch* i, we will prove that protocol Π_{CSN} achieves reliable dissemination *up until epoch* $i+1$. To ensure reliable dissemination, we demonstrate that the diameter of the network of honest nodes is bound by $O(\log n)$. This is achieved by showing that the number of *honest core nodes* is sufficiently large and the connections between honest nodes are *verifiably random*. We then bound the diameter of the network of honest nodes by proving 1) the network of honest core nodes has a connected diameter of $O(\log n)$, and 2) each periphery node is connected to at least one honest core node.

Core-periphery graph. Given the event that protocol Π_{CSN} achieves the security

properties (persistence, liveness, chain growth, chain quality, and consistency) and reliable dissemination up until epoch i, we will prove that protocol Π_{CSN} achieves reliable dissemination up until epoch $i+1$ with high probability.

Let $\mathsf{G} = (\mathsf{N}, \mathsf{E})$ be the network at the i-th epoch[3], where N is the set of participating nodes and E is the set of connections. The network G can be obtained from the *core-periphery graph* that is constructed using the P2P protocol Π_{P2P}. Let $\bar{\mathsf{G}} = (\bar{\mathsf{N}}, \bar{\mathsf{E}})$ be the core-periphery graph, where $\bar{\mathsf{N}} = \mathsf{N} \cup \bar{\mathsf{C}}$, $\bar{\mathsf{C}}$ is a set of core registrations that are included in the registration period, and $\bar{\mathsf{E}}$ is the set of verifiable random connections that are established in the topology construction of the P2P protocol Π_{P2P}. In the P2P protocol Π_{P2P}, each participating node in N acts as *one periphery node* and (possibly) *multiple core nodes*. For example, if a participating node v has two core registrations included in the registration period, it can be mapped as 3 nodes in $\bar{\mathsf{N}}$: one periphery node and two core nodes. Each node in $\bar{\mathsf{G}}$ can be mapped to exactly one node in G. Thus, we can obtain the network G by merging corresponding nodes in the core-periphery graph $\bar{\mathsf{G}}$.

Let $\bar{\mathsf{H}}$ be the set of honest nodes in the core-periphery graph $\bar{\mathsf{G}}$, i.e., the set of core and periphery nodes that are mapped to honest participating nodes in H. Let $\mathsf{G}[\mathsf{H}]$ be the *honest network* with the set of nodes H and the set of edges consisting of all the edges in G that have both endpoints in H. Similarly, let $\bar{\mathsf{G}}[\bar{\mathsf{H}}]$ be the *honest core-periphery graph*. The diameter of the honest network $\mathsf{G}[\mathsf{H}]$ can be upper-bounded by the diameter of the honest core-periphery graph $\bar{\mathsf{G}}[\bar{\mathsf{H}}]$, i.e., $\mathsf{Diam}(\mathsf{G}[\mathsf{H}]) \leq \mathsf{Diam}(\bar{\mathsf{G}}[\bar{\mathsf{H}}])$. Indeed, each node in $\bar{\mathsf{G}}[\bar{\mathsf{H}}]$ can be mapped to exactly one node in $\mathsf{G}[\mathsf{H}]$. Thus, the distance between two honest nodes in $\mathsf{G}[\mathsf{H}]$ is bounded by the distance of the two corresponding nodes in $\bar{\mathsf{G}}[\bar{\mathsf{H}}]$. We show that protocol Π_{CSN} achieves reliable

[3] G is constructed at the end of epoch i and will be used in epoch $i+1$.

dissemination based on the diameter $\mathsf{Diam}(\bar{\mathsf{G}}[\bar{\mathsf{H}}])$ of the honest core-periphery graph as follows.

Lemma 14. *For any epoch i, we have*

$$\Pr\left[\mathcal{R}_{i+1} \mid \mathcal{S}_i \wedge \mathcal{R}_i\right] \geq \Pr\left[\mathsf{Diam}(\bar{\mathsf{G}}[\bar{\mathsf{H}}]) \leq O(\log n) \mid \mathcal{S}_i \wedge \mathcal{R}_i\right].$$

Proof. We first show that any honest node can deliver any message to all other honest nodes within Δ rounds via the honest network $\mathsf{G}[\mathsf{H}]$. Indeed, a node can send messages to all its neighbors on the honest network $\mathsf{G}[\mathsf{H}]$ within δ rounds. Additionally, the distance between two honest nodes on $\mathsf{G}[\mathsf{H}]$ is at most $\frac{\Delta}{\delta}$. Thus, any message from an honest node can be delivered to all honest nodes within Δ rounds.

Next, we show that nodes can finish establishing connections before epoch $i+1$ starts. Equivalently, we will show that nodes cannot generate more than k blocks in t_{e} rounds.

Let r_1 be the first round in the execution in which the length of the chain of any honest node is at least $i \cdot L - k$, i.e., all honest nodes enter the second part of the grace period of epoch i and start establishing the new connection. Let $r_2 = r_1 + t_{\mathrm{e}}$. From round r_1+1 to round r_2, nodes make at most $t_{\mathrm{e}} \cdot Q$ queries to the random oracle in which the probability of success in generating a new block of a single query is p. We represent each query by a Bernoulli random variable with an expected value of p. Let n_{b} be the number of blocks that are generated from round $r_1 + 1$ to round r_2. For any $\epsilon > 0$ such that $t_{\mathrm{e}} < \frac{k}{(1+\epsilon)Q \cdot p}$, using the Chernoff bound in Lemma 56 on at most $t_{\mathrm{e}} \cdot Q$ Bernoulli random variables, we have,

$$\Pr[n_{\mathrm{b}} < k] = 1 - e^{-\Omega(\kappa)}.$$

Thus, at round r_2, with a probability of $1 - e^{-\Omega(\kappa)}$, the length of the longest chain is at most $i \cdot L - k + n_{\mathrm{b}} < i \cdot L$, i.e., nodes have not yet entered epoch $i + 1$. Hence, the network $\bar{\mathsf{G}}^{(i)}$ is constructed before epoch $i + 1$ starts. Additionally, any honest node can deliver any message to all other honest nodes within Δ rounds via the honest network $\mathsf{G}[\mathsf{H}]$. Therefore, protocol Π_{CSN} achieves Δ-reliable dissemination until epoch $i + 1$.

□

Honest core nodes. We demonstrate that the number of honest core nodes is sufficiently large. The selection of core nodes through a PoW mechanism implies that the number of honest core nodes is proportional to the total mining power of honest nodes. Therefore, a big enough core width can ensure that the number of honest core nodes is high enough. To be more specific, in each round, malicious nodes can query the random oracle at most ρQ times, while honest nodes can query it at least $(1 - \rho)$ times. According to Eq. 2.2, the probability of successfully generating a core registration with a single query to the random oracle is $p' = \frac{\mathtt{T}_{\mathrm{core}}}{2^\kappa}$. Thus, the expected number of core registrations of honest nodes in each round is at least $(1 - \rho)Q \cdot p'$. Also, a fraction ω of honest nodes are willing to participate as core nodes. Therefore, in each round, the expected number of core registrations generated by honest nodes willing to participate as core nodes is $(1 - \rho)\omega \cdot Q \cdot p' = (1 - \rho)\omega \cdot Q \cdot p\frac{s}{L_{\mathtt{reg}}}$ (as $p = \frac{\mathtt{T}}{2^\kappa}$ and $\mathtt{T}_{\mathrm{core}} = \frac{s}{L_{\mathtt{reg}}}\mathtt{T}$, we replace $p' = p\frac{s}{L_{\mathtt{reg}}}$).

Nodes can participate as core nodes in the next epoch if their core registrations are included in the registration period of $L_{\mathtt{reg}}$ blocks. From the chain quality property, there is at least one honest block in the last k blocks. Thus, all honest core registrations generated within the first $L_{\mathtt{reg}} - k$ blocks (before the last honest block is generated) will be included in the registration period. It takes nodes at

least $(L_{\mathtt{reg}} - k)(1 - o(1))/(Q \cdot p)$ time to generate $L_{\mathtt{reg}} - k$ blocks. Therefore, in each epoch, the number of honest core nodes is at least $(L_{\mathtt{reg}} - k)(1 - o(1))/(Q \cdot p) \cdot (1 - \rho)\omega \cdot Q \cdot p\frac{s}{L_{\mathtt{reg}}} = \mathsf{q} \cdot s$, where $\mathsf{q} = (1 - o(1))(1 - \rho)\omega(1 - \frac{k}{L_{\mathtt{reg}}})$.

Lemma 15 (Honest core nodes)**.** *Consider any epoch i, let h_{c} be the number of honest core nodes. We have,* $\Pr[h_{\mathrm{c}} \geq \mathsf{q} \cdot s \mid \mathcal{S}_i \wedge \mathcal{R}_i] > 1 - e^{-\Omega(\kappa)}$, *where* $\mathsf{q} = (1-\rho)\omega(1-\frac{1}{L_{\mathtt{reg}}}) - \epsilon$, *and* $\epsilon > 0$.

Proof. We first show that with high probability, all honest nodes have the same view of the set of core nodes. Here, we define the set of core nodes $\bar{\mathsf{C}}$ as the union of all the sets of core nodes in the view of honest nodes. For each participating honest node $u \in \mathsf{H}$, let $\bar{\mathsf{C}}^{(u)}$ be the set of core nodes in the view of node u. The set of core nodes is defined as

$$\bar{\mathsf{C}} = \bigcup_{u \in \mathsf{H}} \bar{\mathsf{C}}^{(u)}.$$

In the CoSpaN protocol, nodes start constructing the network after k blocks since the registration period ends. Thus, based on the consistency property, when nodes construct the network, all honest nodes have the same view of the blocks in the registration period. Hence, they can obtain the same set of core nodes $\bar{\mathsf{C}}$. Consequently, we have that all honest nodes have the same view of the set of core nodes $\bar{\mathsf{H}}_{\mathrm{c}}$.

Let r_1 be the first round in the execution in which the length of the chain of any honest node is at least iL, i.e., all honest nodes are at epoch i.

For any $\epsilon_1 > 0$, let $r_2 = r_1 + (1 - \epsilon_1)\frac{L_{\mathtt{reg}} - k}{Q \cdot p}$. From round $r_1 + 1$ to round r_2, nodes make at most $(1 - \epsilon_1)\frac{L_{\mathtt{reg}} - k}{p}$ queries to the random oracle in which the probability of success in generating a new block of a single query is p. We represent each query by a Bernoulli random variable with an expected value of p. Let n_{b} be the number of blocks that are generated from round $r_1 + 1$ to round r_2. By using the Chernoff

bound in Lemma 56 on at most $(1-\epsilon_1)\frac{L_{\texttt{reg}}-k}{Q\cdot p}$ Bernoulli random variables, we have,

$$\Pr[n_{\text{b}} < k] \leq 1 - e^{-\Omega(\kappa)}.$$

Using the Chernoff bound in Lemma 56, we have,

$$\begin{aligned} &\Pr[n_{\text{b}} < (1-\epsilon_1)(1+\epsilon_1)(L_{\texttt{reg}}-k)]1 - e^{-\Omega(\kappa)} \\ \Rightarrow &\Pr[n_{\text{b}} < (L_{\texttt{reg}}-k)]1 - e^{-\Omega(\kappa)} \end{aligned}$$

Let $\bar{h_{\text{c}}}$ be the number of registration blocks generated by honest nodes willing to participate as core nodes from round r_1 to r_2. In each round, honest nodes willing to participate as core nodes make $(1-\rho)\omega$ queries to the random oracle. Thus, from round r_1 to round r_2, honest nodes willing to participate as core nodes make $(1-\epsilon_1)\frac{L_{\texttt{reg}}-k}{p}(1-\rho)\omega$ queries to the random oracle. Here, the probability of success in generating a new registration block from a single query is $p' = p\frac{s}{L_{\texttt{reg}}}$. We represent each query by a Bernoulli random variable with an expected value of p'. We choose $\epsilon_2 > 0$ such that $1-\epsilon = (1-\epsilon_1)(1-\epsilon_2)$. Using the Chernoff bound in Lemma 56, we have,

$$\Pr[\bar{h_{\text{c}}} > (1-\epsilon)(1-\rho)(1-\frac{1}{L_{\texttt{reg}}})\omega \cdot s] > 1 - e^{-\Omega(\kappa)}.$$

At round r_2, with a probability of 1 - $e^{-\Omega(\kappa)}$, we have, 1) the length of the longest chain is at most $iL + L_{\texttt{reg}} - k$; and 2) the number of registration blocks that are generated by honest nodes, who are willing to participate as core nodes is at least $(1-\epsilon)(1-\rho)(1-\frac{1}{L_{\texttt{reg}}})\omega$.

From the chain quality property in Lemma 12, with a probability of $1 - e^{-\Omega(\kappa)}$, there exists an honest block from the $(i\cdot L + L_{\texttt{reg}} - \kappa + 1)$-th block to the $(i\cdot L + L_{\texttt{reg}})$-th block. Therefore, if an honest node registers as a core node (by propagating a

registration block) before the $(L_{\mathtt{reg}}-k)$-th block of the registration period is generated, it will be selected as a core node in the next epoch (i.e., the registration block will be included on the blockchain). Thus, we have,

$$\Pr[h_{\mathrm{c}} \geq \bar{h_{\mathrm{c}}}] > 1 - e^{-\Omega(\kappa)}$$
$$\Rightarrow \Pr[h_{\mathrm{c}} \geq \mathsf{q} \cdot s] > 1 - e^{-\Omega(\kappa)}$$

□

Verifiable random connections. The verifiable random connections property states that the probability of there being a connection from an honest core node to another honest node (core or periphery) is the same. In CoSpaN protocol, a node core can establish a connection to another node if it can compute a VRF output of the randomness and the other node's identity that is smaller than a given threshold. Based on the consistency property of the blockchain, all nodes will obtain the same randomness value. Additionally, as the output of VRFs is random, we can ensure that the probability of there being a connection from an honest core node to another honest node is the same.

Lemma 16 (Verifiable random connections)**.** *For any honest node $u \in \bar{\mathsf{H}}$ and any honest core node $v \in \bar{\mathsf{H}}_{\mathrm{c}}$, let $\bar{\mu} = \frac{d}{2s} - \varepsilon(\kappa)$, we have,*

$$\Pr[X_{v \to u} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] \geq \bar{\mu}. \tag{2.5}$$

Proof. First, we show that all honest nodes obtain the same randomness rnd_i at epoch i. In the CoSpaN protocol, nodes obtain the randomness rnd_i after k blocks since the registration period ends. Based on the consistency property, at that moment, all honest nodes have the same view of the blocks in the registration period. As the randomness rnd_i is computed based on the last k blocks in the registration period, all

honest nodes obtain the same value of the randomness rnd_i.

Let $\bar{X}_{u\to v}$ be the event where a core node u can solve the inequality in Eq.2.3 for another core or periphery node v. We prove that, for any honest core nodes u, v, we have,

$$\Pr[\bar{X}_{u\to v}] \geq \bar{\mu} = \frac{d}{2s} - \varepsilon(\kappa).$$

Assume towards contradiction that there exists a non-negligible number p such that

$$\Pr[\bar{X}_{u\to v}] < \frac{d}{2s} - p.$$

Let sk' be the private key for VRF of the node u. Since the public-key encryption and the signature schemes are secure, we have,

$$\Pr\left[F_{\mathsf{sk}'}(\mathsf{rnd}_i \| v) < \frac{d}{2s}2^{\kappa}\right] < \frac{d}{2s} - p$$

We will construct an adversary $\mathcal{A}$ that is given a query to the oracle $\mathsf{Prove}_{\mathsf{sk}_v}$ as follows.

- Compute $x = \mathsf{rnd}_i \| v$ and send x to the prover.
- Upon receiving y_b from the prover.
- If $y_b < \frac{d}{2s}2^{\kappa}$, return $b' = 1$. Otherwise, return $b' = 0$.

We have

$$\begin{aligned}\Pr[b = b'] &= \frac{1}{2}\left(\Pr\left[y_0 \geq \frac{d}{2s}2^\kappa\right] + \Pr\left[y_1 < \frac{d}{2s}2^\kappa\right]\right) \\ &= \frac{1}{2}\left(\Pr\left[F_{\mathsf{sk}'}(x) \geq \frac{d}{2s}2^\kappa\right] + \frac{d}{2s}\right) \\ &> \frac{1}{2}\left(1 - \frac{d}{2s} + p + \frac{d}{2s}\right) \\ &= \frac{1}{2}(1+p)\end{aligned}$$

This contradicts the pseudo-randomness property of VRF.

Similarly, for any honest core node u and a periphery node v, we have,

$$\Pr[\bar{X}_{u\to v}] \geq \bar{\mu}.$$

The node v accepts the connection for a core node u if node u can provide an output of the VRF that satisfies the hash inequality in Eq. 2.3. If the node u is honest, it always requests to establish a connection to a node v if it can find the suitable VRF output. Hence, we have,

$$\Pr[X_{u\to v}] = \Pr[\bar{X}_{u\to v}] \geq \bar{\mu}.$$

□

Reliable dissemination. We prove that the Π_{CSN} protocol achieves reliable dissemination by demonstrating that the diameter of the network among honest nodes is bounded by $O(\log n)$. We establish this result in two steps. First, we model the network of honest core nodes as an Erdos-Renyi random network. Following Theorem 7.1 in [45], we can show that the diameter of the network of honest core nodes is bounded by $O(\log n)$. Secondly, each honest core node will connect to any periphery node with a certain probability. Thus, with a sufficient number of core nodes, each

honest periphery node will be connected to at least one honest core node.

Lemma 17. *For $d \geq 4 \log s$, we have,*

$$\Pr[\mathsf{Diam}(\bar{\mathsf{G}}[\bar{\mathsf{H}}_{\mathrm{c}}]) \leq O(\log n)] \geq 1 - e^{-\Omega(\kappa)}.$$

Proof. For any honest core nodes $u, v \in \bar{\mathsf{H}}_{\mathrm{c}}$, we have,

$$A_{uv} \geq \mu,$$

where $\mu = 1 - (1 - \bar{\mu})^2$, $\bar{\mu} = \frac{d}{2s} - \varepsilon(\kappa)$.

Thus, there exists a constant $c_1 > 2$ such that

$$\mu = \frac{c_1 \cdot \log h_{\mathrm{c}}}{h_{\mathrm{c}}}.$$

For any node $v \in \bar{\mathsf{H}}_{\mathrm{c}}$, let

$$N_k(v) = \{w \in \bar{\mathsf{H}}_{\mathrm{c}} : \mathit{dist}(v, w) = k\},$$

where $\mathit{dist}(v, w)$ is the distance between v and w. We define the event

$$F_k = |N_k(v)| \in I_k = \left[\left(\frac{h_{\mathrm{c}} \cdot \mu}{2} \right)^k, (2h_{\mathrm{c}} \cdot \mu)^k \right].$$

For any constant $c' > 0$, let $q = \log(h_{\mathrm{c}}) * \log(h_{\mathrm{c}}^2/c')/\log(h_{\mathrm{c}} \cdot \mu)$. Let $\mathsf{bin}(x, y)$ be the binomial distribution with parameters x and y, i.e., the discrete probability distribution of the number of successes in a sequence of x independent experiments

in which the probability of success in each experiment is y. For $k \leq \lceil q/2 \rceil$, we have

$$\begin{aligned}
&\Pr[\bar{F}_k \mid \ F_1, \cdots, F_{k-1}] \\
=&\Pr\left[\mathsf{bin}\left(h_{\mathrm{c}} - \sum_{i=1}^{k-1} |N_i(v), 1-(1-p)^{|N_{k-1}(v)|}\right) \notin I_k\right] \\
\leq&\Pr\left[\mathsf{bin}\left(h_{\mathrm{c}} - o(h_{\mathrm{c}}), \frac{3}{4}\left(\frac{h_{\mathrm{c}}\cdot\mu}{2}\right)^{k-1}\mu\right) \leq \left(\frac{h_{\mathrm{c}}\cdot\mu}{2}\right)^{k}\right] \\
&\qquad + \Pr\left[\mathsf{bin}\left(h_{\mathrm{c}} - o(h_{\mathrm{c}}), \frac{5}{4}\left(2h_{\mathrm{c}}\cdot\mu\right)^{k-1}\mu\right) \geq \left(2h_{\mathrm{c}}\cdot\mu\right)^{k}\right] \\
&= e^{-\Omega((h_{\mathrm{c}}\cdot\mu)^k)} = e^{-\Omega(d^k)}.
\end{aligned}$$

Using union bound, we have

$$\begin{aligned}
\Pr[\bar{F}_k] &\leq \sum_{i=1}^{k} \Pr[\bar{F}_i \mid \ F_1, \cdots, F_{i-1}] \\
&= \sum_{i=1}^{k} e^{-\Omega(d^k)} = e^{-\Omega(d)}.
\end{aligned}$$

Thus, with probability $1 - e^{-\Omega(d)}$, for any node u, v, we have

$$\begin{aligned}
|N_{\lceil \log(h_{\mathrm{c}})/2 \rceil}(v)| &\geq \left(\frac{h_{\mathrm{c}}\cdot\mu}{2}\right)^{\lceil \log(h_{\mathrm{c}})/2 \rceil}, \\
\text{and } |N_{\lfloor \log(h_{\mathrm{c}})/2 \rfloor}(u)| &\geq \left(\frac{h_{\mathrm{c}}\cdot\mu}{2}\right)^{\lfloor \log(h_{\mathrm{c}})/2 \rfloor}.
\end{aligned}$$

Let $X = N_{\lceil \log(h_{\mathrm{c}})/2 \rceil}(v)$ and $Y = N_{\lfloor \log(h_{\mathrm{c}})/2 \rfloor}(u) \neq \emptyset$ We have, either $X \cap Y \neq \emptyset$ or

$$\begin{aligned}
\Pr[\nexists \text{ an edge between } X, Y] &= (1-\mu)^{\left(\frac{h_{\mathrm{c}}\cdot\mu}{2}\right)^{\log(h_{\mathrm{c}})}} \\
&\leq e^{-\Omega(d)}.
\end{aligned}$$

In other word, for any node u, v, we have

$$\Pr[dist(u,v) > \log(h_{\mathrm{c}}) + 1] \leq e^{-\Omega(d)}.$$

Using union bound on all h_c^2 pairs of nodes, we have,

$$\begin{aligned}\Pr[\mathsf{Diam}(\bar{\mathsf{G}}[\bar{\mathsf{H}}_c]) > \log(h_c) + 1] &\leq h_c^2 e^{-\Omega(d)} = e^{-\Omega(d) - \log h_c} \\ &= e^{-\Omega(d)}.\end{aligned}$$

□

Lemma 18. *Let* RB *be the event where all honest periphery nodes have at least one connection to a core node. For* $d > \frac{k}{-\log \mathsf{q}}$. *We have,*

$$\Pr[\mathsf{RB} \mid \mathcal{S}_i \wedge \mathcal{R}_i] \geq 1 - e^{-\Omega(\kappa)}.$$

Proof. Recall that, the probability that there exists a connection between an honest core and an honest periphery node is $\frac{d}{2s}$. Thus, the probability that an honest periphery node has no connection to any honest core nodes is

$$\begin{aligned}\left(1 - \frac{d}{2s}\right)^{h_c} &= \left(\left(1 - \frac{d}{2s}\right)^{\frac{2s}{d}}\right)^{\frac{h_c \cdot d}{2s}} \\ &\leq e^{-\frac{h_c \cdot d}{2s}}\end{aligned}$$

Since the number of periphery honest node is smaller than h, using union bound, we have, the probability that there exists an honest periphery node has no connection to any honest core nodes is

$$\begin{aligned}n \cdot e^{-\frac{h_c d}{2s}} &= e^{-\left(\frac{h_c d}{2s} - \log h\right)} \\ &= e^{-\Omega(\kappa)}.\end{aligned}$$

□

We are now ready to prove the CoSpaN protocol achieve reliable dissemination.

Lemma 19 (Step i-B)**.** *Let* $\Delta = O(\log n) \times \delta$*,* $k = \Omega(\kappa)$*. Consider protocol* Π_{CSN} *with parameters* $L > \frac{k}{(1-\rho)\omega} + 2k$*,* $s > k$ *and* $d \geq \max(\frac{k}{-\log \mathsf{q}}, 4 \log s)$*. We have,* $\Pr[\mathcal{R}_{i+1} \mid \mathcal{S}_i \wedge \mathcal{R}_i] > 1 - \epsilon(\kappa)$.

Proof. Combining Lemma 17 and Lemma 18, with probability $1 - e^{-\Omega(\kappa)}$, we have that the maximum distance between two honest core nodes is at most $\log h_{\mathrm{c}} + 1$, and all honest periphery nodes connect to at least one honest core node. Thus, the maximum distance between an honest periphery node and an honest core node is at most $\log h_{\mathrm{c}} + 2$. Therefore, the maximum distance between two honest periphery nodes is at most $\log h_{\mathrm{c}} + 3$.

We have that

$$\Pr[\mathsf{Diam}(\bar{\mathsf{G}}^{(i)}) \leq \log h_{\mathrm{c}} + 3] \geq 1 - e^{-\Omega(d)}.$$

Note that an honest node can send any messages to all of its neighbors within a network round of δ_l time units. Thus, with a probability of $1 - e^{-\Omega(d)}$, an honest node can send any messages to all honest nodes within

$$\Delta = O(\log h_{\mathrm{c}}) \times \delta_l.$$

□

2.3.2 Network sparsity

We analyze the sparsity of the CoSpaN network. In the CoSpaN protocol, nodes establish verifiable random connections to other nodes based on public randomness. If the randomness is fixed, we can use a concentration inequality to bound the number of connections in the network. However, an adversary can perform a *randomness grinding attack*, i.e., use different values of randomness to increase the chance of connecting malicious nodes to honest nodes and increasing the sparsity. Fortunately,

based on the chain quality property, the adversary cannot try too many different values of randomness. Thus, the sparsity of the network remains bounded.

Lemma 20 (Network sparsity)**.** *For any $\epsilon > 0$, with probability $1 - e^{\Omega(d)}$, the average degree (sparsity) of an honest node in G is at most $D = \frac{2(1+\epsilon)}{(1-\eta)}(1 + \frac{s}{n})d$.*

Proof. We first analyze the sparsity of the core-periphery graph $\bar{\mathsf{G}}$ and the network G. To analyze the sparsity of the core-periphery graph, we analyze the maximum number of core nodes and then bound the number of connections from those core nodes.

We bound the number of connection in the core-periphery graph when the randomness rnd_i is fixed. Let r_1 be the first round in the execution in which the length of the chain of any honest node is at least iL, i.e., all honest nodes are at epoch i. For any $\epsilon_1 > 0$, let $r_2 = r_1 + (1+\epsilon_1)\frac{L_{\mathtt{reg}}}{Qp}]$. Let n_{b}, n_{c} be the number of blocks, registration blocks that are generated from round $r_1 + 1$ to round r_2.

From round r_1+1 to round r_2, nodes make at most $(1+\epsilon_1)\frac{L_{\mathtt{reg}}-k}{p}$ queries to random oracle in which the probability of success in generating a new block, registration block of a single query are p, p' respectively.

We choose ϵ_2 such that $(1-\epsilon_2)(1+\epsilon_1) = 1$, using the Chernoff bound in Lemma 56 on at most $(1+\epsilon_1)\frac{L_{\mathtt{reg}}-k}{p}$ Bernoulli random variables with the expected value of p, we have,

$$\Pr[n_{\mathrm{b}} \geq L_{\mathtt{reg}} - k] \geq 1 - e^{-\Omega(\kappa)}.$$

We choose $\epsilon_3, \epsilon_4 > 0$ such that $1 + \epsilon_4 = (1 + \epsilon_1)(1 + \epsilon_3)$. Using the Chernoff bound in Lemma 56 on at most $(1 + \epsilon_1)\frac{L_{\mathtt{reg}}-k}{p}$ Bernoulli random variables with the

expected value of $p' = p\frac{s}{L_{\mathtt{reg}}}$, we have,

$$\Pr[n_{\mathrm{b}} \geq (1+\epsilon_4)s] \geq 1 - e^{-\Omega(\kappa)}.$$

At round r_2, with a probability of 1 - $e^{-\Omega(\kappa)}$, we have, 1) the length of the longest chain is at least $iL + L_{\mathtt{reg}}$; and 2) the number of registration blocks is at most $(1+\epsilon_4)s$. Recall that, in the CoSpaN protocol, nodes can only register as core nodes in the registration period. Thus, with a probability of 1 - $e^{-\Omega(\kappa)}$, the number of core nodes at epoch i is at most $\hat{s} = (1+\epsilon_4)s$.

For a core node v and a (core or periphery) node u, let $X_{v\to u}$ be the Bernoulli random variable that represent whether or not there exists a connection from v to u. We have, $\Pr[X_{v\to u} = 1] = \bar{\mu} = \frac{d}{2s}$.

In the the core-periphery graph $\bar{\mathsf{G}}$, with a probability of 1 - $e^{-\Omega(\kappa)}$, there are at most n periphery nodes and $\hat{s}$ core nodes. Thus, the number of connections in G_+ is at most

$$\sum_{v=1}^{\hat{s}} \sum_{u=1}^{n+\hat{s}} X_{v\to u}.$$

Using the Chernoff bound in Lemma 56, for $\epsilon_5 > 0$, we have,

$$\Pr[\sum_{v=1}^{\hat{s}} \sum_{u=1}^{n} X_{v\to u} \leq (1+\epsilon_5)\hat{s}(n+\hat{s})\frac{d}{2s}] \geq 1 - e^{-\Omega((n+s)\frac{d}{2})}.$$

Hence, there exists a constant $\epsilon > 0$ such that, with a probability of 1 - $e^{-\Omega(\kappa)}$, the number of connections in G_+ is at most $(1+\epsilon)(n+s)\frac{d}{2}$.

Now, we analyze degree of a node when the randomness rnd is computed based on the hash value of the blocks in the previous epoch. Since the miner does not know the hash value until the new block is created, we consider the hash value of *any block* as a perfect randomness. In other words, the adversary cannot predict the hash value

of future blocks (even when that block is create by a malicious miner). Thus, the adversary can only manipulate by changing the hash value of the last block in the registration period, i.e., if an malicious miner create the last block in the registration period, the adversary can choose either to publish the that block or not. Thus, the adversary can try at most $O(\kappa)$ different values of the randomness rnd. Using union bound, we have with probability at least $1 - \kappa e^{\Omega((n+s)\frac{d}{2} - \log k)} = 1 - e^{\Omega(\kappa)}$, for any constant $\epsilon > 0$, the number of connections in G_+ is at most $(1+\epsilon)(n+s)\frac{d}{2}$.

We now bound the sparsity of the network G based on the core-periphery graph $\bar{\mathsf{G}}$. Since the sum of degree of nodes in network G does not exceed that in the the core-periphery graph $\bar{\mathsf{G}}$, with a probability of 1 - $e^{-\Omega(\kappa)}$, the number of connections in G is at most $(1+\epsilon)(n+s)\frac{d}{2}$. Thus, the sum of degree of nodes in G is

$$(1+\epsilon)(n+s)d.$$

As the number of honest nodes is at least $n(1-\eta)$, with a probability of 1 - $e^{-\Omega(\kappa)}$the average degree of nodes in G is

$$(1+\epsilon)\frac{(n+s)d}{n(1-\eta)}(1+\epsilon) = (1+\frac{s}{n})/(1-\eta)d.$$

□

We also analyze the degree of an honest physical node. We consider two cases of mining nodes and non-mining nodes as follows.

- *Degree of a mining node.* Let q be the maximum number of queries that an honest node can make in a round. For any $\epsilon > 0$, with high probability, an honest node can register at most $(1+\epsilon)\frac{q}{Q}s$ times as core nodes. Thus, for any $\epsilon > 0$, the degree of an honest physical node is at most $(1+\epsilon)\frac{q(2s+n)}{2Q}d$.

- *Degree of a non-mining node.* A non-mining node can only participate and

periphery node. Thus, the average degree of a non-mining node is $\frac{d}{2}$.

2.3.3 Main theorem

We are now ready to prove the security of CoSpaN protocol in a sparse network by induction as follows. From the bootstrapping condition and Lemma 13, we have protocol Π_{CSN} is secure in the bootstrapping epoch. Then, combining Lemma 13 and Lemma 19, if protocol Π_{CSN} is secure in epoch $i-1$, it is secure in epoch i.

Theorem 21 (Static security in a sparse network)**.** *Consider* Γ^*_{stat}*-admissible environments and protocol* Π_{CSN} *with parameters* $L > \frac{k}{(1-\rho)\omega} + 2k$*,* $s > k$ *and* $d \geq \max(\frac{k}{-\log \mathsf{q}}, 4\log s)$*. Under the bootstrapping condition is satisfied, protocol* Π_{CSN} *achieves*

- *persistence, liveness ; and*
- *D-sparsity, where* $D = 2(1+\epsilon)(1+\frac{s}{n})/(1-\eta)d$*.*

Proof. We prove the security of protocol Π_{CSN}, i.e.,

$$\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa),$$

by induction. For each epoch $j \in [1..e_{\max}]$, we prove the two following hypothesis

$$\Pr[\mathcal{S}_j \wedge \mathcal{R}_{j+1}] > 1 - (2j+1)\varepsilon(\kappa).$$

As the number of epoch in the protocol execution is at most $e_{\max}$, we can conclude that protocol Π_{CSN} is secure if the hypothesis is true for $j = 2e_{\max}$, i.e.,

$$\Pr[\mathcal{S}_{e_{\max}} \wedge \mathcal{R}_{e_{\max}+1}] > 1 - (2e_{\max}+1)\varepsilon(\kappa).$$

Basis step. From the bootstrapping condition, protocol Π_{CSN} achieve reliable dissem-

ination at epoch 1, i.e.,

$$\Pr[\mathcal{R}_1] \geq 1 - \varepsilon(\kappa).$$

Induction step. Given that the hypothesis is true for $j = i - 1$, (where $i \in [1..e_{\max}]$), we will prove the hypothesis is true of $j = i$. The induction step consists of two parts as follows.

- *Step i-A:* Given from step $(i-1)$-B that

  $$\Pr[\mathcal{S}_{i-1} \wedge \mathcal{R}_i] > 1 - (2i-1)\varepsilon(\kappa).$$

  From Lemma 13, we have,

  $$\begin{aligned}\Pr[\mathcal{S}_i \wedge \mathcal{R}_i] &\geq \Pr[\mathcal{S}_i \wedge \mathcal{R}_i \mid \mathcal{S}_{i-1} \wedge \mathcal{R}_i] \times \Pr[\mathcal{S}_{i-1} \wedge \mathcal{R}_i] \\ &= \Pr[\mathcal{S}_i \mid \mathcal{S}_{i-1} \wedge \mathcal{R}_i] \times \Pr[\mathcal{S}_{i-1} \wedge \mathcal{R}_i] \\ &> (1 - (2i-1)\varepsilon(\kappa))(1 - \varepsilon(\kappa)) \\ &> 1 - 2i\varepsilon(\kappa).\end{aligned}$$

- *Step i-B:* Given from step i-A that

  $$\Pr[\mathcal{S}_i \wedge \mathcal{R}_i] > 1 - 2i\varepsilon(\kappa).$$

  From Lemma 19, we have,

  $$\begin{aligned}\Pr[\mathcal{S}_i \wedge \mathcal{R}_{i+1}] &\geq \Pr[\mathcal{S}_i \wedge \mathcal{R}_{i+1} \mid \mathcal{S}_i \wedge \mathcal{R}_i] \times \Pr[\mathcal{S}_i \wedge \mathcal{R}_i] \\ &= \Pr[\mathcal{R}_{i+1} \mid \mathcal{S}_i \wedge \mathcal{R}_i] \times \Pr[\mathcal{S}_1 \wedge \mathcal{R}_i] \\ &> (1 - 2i\varepsilon(\kappa))(1 - \varepsilon(\kappa)) \\ &> 1 - (2i+1)\varepsilon(\kappa).\end{aligned}$$

Thus, we have,

$$\begin{aligned}&\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}_i) = 1]\\ \geq & 1 - 2i\varepsilon(\kappa).\end{aligned}$$

Since the number of rounds in $\mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa)$ is polynomial in κ, the number of epochs in $\mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa)$ is also polynomial in κ. In other words, there exists a polynomial $\mathsf{poly}()$ such that $e_{\max} = \mathsf{poly}(\kappa)$. We have

$$\begin{aligned}&\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}) = 1]\\ = & \Pr[\text{VIEW} \leftarrow \mathsf{EXEC}^{\Pi_{\text{CSN}}}_{\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{sec}(\text{VIEW}_{e_{\max}}) = 1]\\ \geq & 1 - 2e_{\max}\varepsilon(\kappa) = 1 - 2\mathsf{poly}(\kappa)\varepsilon(\kappa) = 1 - \varepsilon'(\kappa),\end{aligned}$$

where $\varepsilon'(\cdot)$ is a negligible function.

Further, from Lemma 20, the average degree of a node in each epoch is at most D. Recall that, each epoch consists of L blocks. Thus, for any $\epsilon > 0$, with probability $1 - e^{\Omega(\kappa)}$, each epoch lasts for at least $t_{\text{s}} = \frac{L}{(1+\epsilon)(\alpha+\beta)}$ rounds. In t_{s} rounds, the length of the chain of honest players increase by at most L. Thus, for any t_{s} consecutive rounds, there is no overlap between the connections in $\mathsf{G}^{(j)}$ and $\mathsf{G}^{(j+2)}$ ($j \in \mathbb{N}$). Hence, protocol Π_{CSN} achieves $2D$-sparsity. □

2.4 Weakly Adaptive security

In this section, we demonstrate the security of the CoSpaN protocol in several (weakly) adaptive settings. In Subsection 2.4.1, we consider an *M-adaptive adversary* that requires some time to corrupt honest nodes. Then, in Subsection 2.4.2, we examine an *S-adaptive adversary* that can instantly corrupt honest nodes. In Appendix 2.4.3, we conduct further analysis on the security of the protocol in the presence of an

S-adaptive adversary, particularly when the mining power is concentrated on a few top nodes.

2.4.1 M-adaptive security

We extend the model in Section 5.1 in the presence of a *mildly adaptive* (M-adaptive) adversary that can corrupt honest nodes during the protocol execution in which the time it takes to corrupt honest nodes is τ_{corr} rounds. We can prove that protocol Π_{CSN} achieves the same security properties as in the static setting (Section 2.3).

Protocol Π_{CSN} can defend against an M-adaptive adversary due to the fact that *the network topology dynamically changes* in every epoch. According to the chain growth property, within τ_{corr} rounds, the length of the chain of honest players increases by at least $2L$ blocks. In other words, it takes at least 2 epochs for the adversary to corrupt honest nodes. In order to corrupt some honest nodes at epoch i, the adversary must begin the corruption before epoch $i-2$. Since the core nodes of epoch i are selected at epoch $i-1$, the adversary *cannot predict the network topology* of epoch i at epoch $i-2$. Therefore, the adversary cannot gain an advantage by adaptively corrupting the honest nodes.

We consider admissible environments in the presence of an M-adaptive adversary with a predicate $\Gamma^{*}_{\text{M-adap}}$ in which the honest majority assumption is given and $\tau_{\text{corr}} > 2\left(\frac{k}{(1-\rho)\omega} + 2k\right) / \left((1-\epsilon)\,\alpha\right)$.

Definition 22 ($\Gamma_{\text{M-adap}}$-admissible environments)**.** *We say a tuple* $(n, Q, \eta, \rho, \delta, \tau_{\text{corr}}, \mathcal{A}, \mathcal{Z})$ *with* $\Gamma_{\text{M-adap}}(n, Q, \eta, \rho, \delta, \tau_{\text{corr}}) = 1$ *is* $\Gamma_{\text{M-adap}}$*-admissible w.r.t* $(\Pi^{\mathsf{V}}, \Lambda)$ *if* $\mathcal{A}$*,* $\mathcal{Z}$ *are probabilistic polynomial-time algorithms, and for every* VIEW *in the support of* $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$*, the following holds:*

1. *In each round, the environment* $\mathcal{Z}$ *can adaptively corrupt additional nodes as*

long as the number and the hash power of malicious nodes do not exceed ηn and ρQ, respectively. The corruption of an honest node will take τ_{corr} rounds to complete.

2. *Conditions (2), (3), (4) in Definition 1.*

Theorem 23 (M-adaptive security)**.** *Let $k = \Omega(\kappa)$. Consider $\Gamma^{*}_{\text{M-adap}}$-admissible environments and protocol Π_{CSN} with parameters $\frac{k}{(1-\rho)\omega} + 2k < L < \frac{\tau_{\text{corr}}(1-\epsilon)\alpha}{2}$, $\epsilon \in (0,1)$, $s > k$ and $d \geq \max(\frac{k}{-\log \mathsf{q}}, 4\log s)$. Under the bootstrapping condition, protocol Π_{CSN} achieves persistence and liveness; and sparsity.*

Proof. From the chain growth property, in τ_{corr} rounds, the length of the chain of honest players increases by at least $2L$ blocks. In other words, it takes at least 2 epochs for the adversary to corrupt honest nodes. Thus, to corrupt some honest nodes at epoch i, the adversary must start the corruption before epoch $i-2$. In the execution of protocol Π_{CSN}, the network topology is unpredictable until the core nodes are selected. As the core nodes of epoch i are selected at epoch $i-1$, the adversary cannot predict the network topology of epoch i at epoch $i-2$. Hence, we can achieve the same security as in the static case in Theorem 21. □

2.4.2 S-adaptive security

We analyze the security of the CoSpaN protocol against a *slow-observation adaptive* (S-adaptive) adversary that can instantly corrupt honest nodes, but takes τ_{obs} rounds to observe connections among honest nodes. The S-adaptive adversary is stronger than the M-adaptive adversary when $\tau_{\text{obs}} = \tau_{\text{corr}}$. The M-adaptive adversary needs to start corrupting some honest nodes at round $r - \tau_{\text{corr}}$ to corrupt them at round r, while the S-adaptive adversary can start at round r. Therefore, the S-adaptive adversary can gather more information to decide which nodes to corrupt.

To defend against an S-adaptive adversary, we need to increase the core width, which in turn increases the network sparsity.

Restricted point-to-point communication model. To allow a slow observation of connections, we introduce a restricted point-to-point communication model in which the environment is responsible for delivering all messages. The deliver delay is determined by an algorithm $\mathcal{D}$ that is proposed by the adversary $\mathcal{A}$ before the execution starts.

Delivering message. Consider at round r, an honest node u send a message to a neighbor v. At a round $r' \in [r+1..r+\delta]$, the environment $\mathcal{Z}$ takes the view of all nodes at the current round and uses algorithm $\mathcal{D}$ to determine whether or not to send the message to node v at the current round. At round $r+\delta$, if the node v have not received the message yet, the environment $\mathcal{Z}$ will send the message to node v.

Observing connection. The adversary can observe a connection after τ_{obs} rounds. Specifically, consider a connection between two nodes u and v that is established at round r. At round $r+\tau_{\text{obs}}$, if u and v are still connected, the adversary knows about the existence of the connection between u and v.

We consider admissible environments in the presence of S-adaptive adversaries with a predicate $\Gamma^{*}_{\text{S-adap}}$ in which the honest majority assumption is given and $\tau_{\text{obs}} > \frac{4k}{\alpha} + \frac{2k}{(1-2\rho)\omega\cdot\alpha}$.

Definition 24 ($\Gamma_{\text{S-adap}}$-admissible environments)**.** *We say a tuple* $(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}, \mathcal{A}, \mathcal{Z})$ *with* $\Gamma_{\text{S-adap}}(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}) = 1$ *is* $\Gamma_{\text{S-adap}}$*-admissible w.r.t* $(\Pi^{\mathsf{V}}, \Lambda)$ *if* $\mathcal{A}$*,* $\mathcal{Z}$ *are probabilistic polynomial-time algorithms, and for every* VIEW *in the support of* $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$*, the following holds:*

1. *In each round, the environment* $\mathcal{Z}$ *can adaptively instantly corrupt additional nodes as long as the number and the hash power of malicious nodes do not exceed*

ηn and ρQ, respectively.

2. *Conditions (2), (3) in Definition 1.*

3. *The adversary can observe new connections among nodes after τ_{obs} rounds.*

We set the core width to ensure that most honest nodes are selected as core nodes. This way, even if the adversary can corrupt a set of nodes that control a fraction ρ of mining power, we can still guarantee a large enough number of honest core nodes.

Lemma 25 (Honest core nodes in S-adaptive security)**.** *For any $\epsilon > 0$, let $\mathsf{q}_{\text{M-adap}} = (1-2\rho)\omega - \frac{k}{L_{\text{reg}}} - \epsilon$. Consider protocol Π_{CSN} with parameter $s > k + \frac{2(1+\epsilon)}{\mathsf{q}_{\text{M-adap}} \cdot \epsilon^2 \log_2 e} h$, we have,*

$$\Pr[h_{\text{c}} \geq s \cdot \mathsf{q}_{\text{M-adap}} \mid \mathcal{S}_i \wedge \mathcal{R}_i] > 1 - e^{-\Omega(\kappa)}.$$

Proof of Lemma 25. Let H be the set of honest nodes. Consider a set $W \subseteq \mathsf{H}$ in which the total mining power of all nodes in W is at least $1-2\rho$.

Let r_1 be the first round in the execution in which the length of the chain of any honest node is at least iL, i.e., all honest nodes are at epoch i.

For any $\epsilon_1 > 0$, let $r_2 = r_1 + (1-\epsilon_1)\frac{L_{\text{reg}}-k}{Q \cdot p}$. From round $r_1 + 1$ to round r_2, nodes make at most $(1-\epsilon_1)\frac{L_{\text{reg}}-k}{p}$ queries to random oracle in which the probability of success in generating a new block of a single query is p. We represent each query by a Bernoulli random variable with an expected value of p. Let n_{b} be the number of blocks that are generated from round $r_1 + 1$ to round r_2. By using the Chernoff bound in Lemma 56 on at most $(1-\epsilon_1)\frac{L_{\text{reg}}-k}{Q \cdot p}$ Bernoulli random variables, we have,

$$\Pr[n_{\text{b}} < k] 1 - e^{-\Omega(\kappa)}.$$

Using the Chernoff bound in Lemma 56, we have,

$$\Pr[n_{\mathrm{b}} < (1-\epsilon_1)(1+\epsilon_1)(L_{\mathtt{reg}} - k)]1 - e^{-\Omega(\kappa)},$$
$$\Rightarrow \Pr[n_{\mathrm{b}} < (L_{\mathtt{reg}} - k)]1 - e^{-\Omega(\kappa)}.$$

Let $\bar{h_{\mathrm{c}}}$ be the number of registration blocks that are generated by the honest nodes that are willing to participate as core nodes from round r_1 to r_2. In each round, the honest nodes in W make at least $(1-2\rho)\omega$ queries to the random oracle. Thus, from round r_1 to round r_2, the honest nodes in W make at least $(1-\epsilon_1)\frac{L_{\mathtt{reg}}-k}{p}(1-2\rho)\omega$ queries to the random oracle. Here, the probability of success in generating a new registration block of a single query is $p' = p\frac{s}{L_{\mathtt{reg}}}$. We represent each query by a Bernoulli random variable with an expected value of p'. We choose $\epsilon_2 > 0$ such that $1-\epsilon = (1-\epsilon_1)(1-\epsilon_2)$. Using the Chernoff bound in Lemma 56, we have,

$$\Pr[\bar{h_{\mathrm{c}}} > (1-\epsilon)(1-2\rho)(1-\frac{1}{L_{\mathtt{reg}}})\omega s] > 1 - e^{-\Omega(\kappa)}.$$

At round r_2, with a probability of 1 - $e^{-\Omega(\kappa)}$, we have, 1) the length of the longest chain is at most $iL + L_{\mathtt{reg}} - k$; and 2) the number of registration blocks that are generated by honest nodes in W who are willing to participate as core nodes is at least $(1-\epsilon)(1-2\rho)(1-\frac{1}{L_{\mathtt{reg}}})\omega$.

From chain quality property in Lemma 12, with probability $1 - e^{-\Omega(\kappa)}$, there exist an honest block from the $(iL + L_{\mathtt{reg}} - \kappa + 1)$-th blocks to the $(iL + L_{\mathtt{reg}})$-th blocks. Thus, if an honest node register as a core node (by propagating a registration block) before the $(L_{\mathtt{reg}} - k)$-th block of the registration period is generated, it will be selected as a core node in the next epoch (i.e., the registration block is included on

the blockchain). Thus, we have,

$$\Pr[h_{\mathrm{c}} \geq \bar{h_{\mathrm{c}}}] > 1 - e^{-\Omega(\kappa)},$$
$$\Rightarrow \Pr[h_{\mathrm{c}} \geq \mathsf{q}_{\text{M-adap}} \cdot s] > 1 - e^{-\Omega(\kappa)}.$$

□

Next, we demonstrate that the connections between honest core nodes and other honest nodes are verifiably random and unpredictable. This means the adversary has no knowledge of whether there are connections among honest nodes.

Lemma 26 (Private connection in S-adaptive security)**.** *Let* $\bar{\mathsf{H}}$ *be the set of honest (core and periphery) nodes and* $\bar{\mathsf{H}}_{\mathrm{c}} \subseteq \bar{\mathsf{H}}$ *be the set of honest core nodes at the beginning of epoch* i*. Until some round* r *at epoch* i*, let* $\bar{\mathsf{W}} \subseteq \bar{\mathsf{H}}$ *be the set of nodes that is corrupted by the adversary. Let* $\bar{\mathsf{W}}_{\mathrm{c}} \subseteq \bar{\mathsf{W}}$ *be the set of corrupted core nodes. Consider an adversary* $\mathcal{A}$ *runs some* PPT *algorithm to return an honest node* $u \in \bar{\mathsf{H}} \setminus \bar{\mathsf{W}}$ *and an honest core node* $v \in \bar{\mathsf{H}}_{\mathrm{c}} \setminus \bar{\mathsf{W}}_{\mathrm{c}}$*. We have,*

$$\Pr[X_{v \to u} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] \leq \bar{\mu} + \varepsilon(\kappa). \tag{2.6}$$

Proof of Lemma 26. We consider two cases of node u as follows.

The node u *is a core node (core-core connection).* Assume toward contradiction that there exists an adversary $\mathcal{A}$ that can break the security definition in Eq. 2.6. Let $\mathsf{sk}', \mathsf{pk}'$ be the key pair that the node v holds. Recall that, the node v broadcast an encryption using the public key of the node u. Hence, the adversary cannot learn any information, including the VRF output, from the encryption.

We construct the adversary $\mathcal{A}'$ that is given query to the oracle $\mathsf{Prove}_{\mathsf{sk}'}$ as follows.

- Run $\mathcal{A}$ and obtain u and v from $\mathcal{A}$.
- Compute $x = \mathsf{rnd}_i \| u$ and send x to the prover.

- Upon receiving y_b from the prover.
- If $y_b < 2^\kappa \frac{d}{2s}$, return $b = b'$. Otherwise, return $b = 1 - b'$.

We have,

$$\begin{aligned}
&\Pr[b \neq X_{v\to u}] \\
&= \frac{d}{2s}\Pr[b = 0 | X_{v\to u} = 0] + (1 - \frac{d}{2s})\Pr[b = 1 | X_{v\to u} = 1] \\
&= \frac{d}{2s}\Pr[b = 0 | F_{\mathsf{sk}_v}(x) < \frac{d}{2s}2^\kappa] \\
&\quad + (1 - \frac{d}{2s})\Pr[b = 1 | F_{\mathsf{sk}_v}(x) \geq \frac{d}{2s}2^\kappa].
\end{aligned}$$

Recall that, as $\mathcal{A}$ that can break the security definition in Eq. 2.6, there exists a non-negligible number p such that

$$\begin{aligned}
&\Pr[X_{v\to u} = 1] > \frac{d}{2s} + p, \\
\Rightarrow &\Pr[F_{\mathsf{sk}_v}(\mathsf{rnd}_i \| v) < 2^\kappa \frac{d}{2s}] > \frac{d}{2s} + p.
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\Pr[b = b'] &= \frac{1}{2}\Pr[F_{\mathsf{sk}_v}(\mathsf{rnd}_i \| v) < 2^\kappa \frac{d}{2s} | b = 0] + \\
&\qquad \frac{1}{2}\Pr[F_{\mathsf{sk}_v}(\mathsf{rnd}_i \| v) \geq 2^\kappa \frac{d}{2s} | b = 1] \\
&\geq \frac{1}{2}\left(\frac{d}{2s} + p\right) + \frac{1}{2}\left(1 - \frac{d}{2s}\right) \\
&= \frac{1}{2} + \frac{1}{2}p.
\end{aligned}$$

This contradicts the pseudorandomness property of VRF.

The node u is a periphery node (core-periphery connection). Here, the v directly send a request to connect to node u. Thus, the adversary cannot learn the VRF output that node v sends to node u. Similar to the core-core connection, can prove the core-periphery connection between two honest nodes is private.

□

Private connections ensure that an adversary cannot take advantage by adaptively corrupting honest nodes. This ensures that connections remain verifiable random, even in the presence of an S-adaptive adversary.

Lemma 27 (Random verifiable connection in S-adaptive security)**.** *For any honest node* $u \in \bar{\mathsf{H}} \setminus \bar{\mathsf{W}}$ *and any honest core node* $v \in \bar{\mathsf{H}}_{\mathrm{c}} \setminus \bar{\mathsf{W}}_{\mathrm{c}}$*, we have,*

$$\Pr[X_{v\to u} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] \geq \bar{\mu}. \tag{2.7}$$

Proof. We assume toward contradiction that there exists an honest node $u \in \bar{\mathsf{H}} \setminus \bar{\mathsf{W}}$ and an honest core node $v \in \bar{\mathsf{H}}_{\mathrm{c}} \setminus \bar{\mathsf{W}}_{\mathrm{c}}$ and a non-negligible number p such that

$$\Pr_{v\leftarrow\bar{\mathsf{H}}_{\mathrm{c}}\setminus\bar{\mathsf{W}}_{\mathrm{c}},u\leftarrow\bar{\mathsf{H}}\setminus\bar{\mathsf{W}}}[X_{v\to u} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] < \frac{d}{2s} - p.$$

Recall from Lemma 16 that

$$\Pr_{v\leftarrow\bar{\mathsf{H}}_{\mathrm{c}},u\leftarrow\bar{\mathsf{H}}}[X_{v\to u} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] \geq \bar{\mu}.$$

Thus, there exists a pair of nodes $v' \in \bar{\mathsf{H}}_{\mathrm{c}}, u' \in \bar{\mathsf{H}}$ in which $v' \in \bar{\mathsf{W}}_{\mathrm{c}}$ and $u' \in \bar{\mathsf{W}}$ such that

$$\Pr[X_{v'\to u'} = 1 \mid \mathcal{S}_i \wedge \mathcal{R}_i] > \frac{d}{2s} + p',$$

where p' is a non-negligible number. This contradicts the private connection property in Lemma 26.

□

Now, we can follow the same proof in the Lemma 19 (except replacing the core quality q by $\mathsf{q}_{\text{M-adap}}$) to prove that protocol Π_{CSN} achieves Δ-reliable dissemination. Then, following the same proofs and Theorem 21, we can prove the security of protocol Π_{CSN} in the presence of an S-adaptive adversary.

Theorem 28 (S-adaptive security)**.** *Consider* $\Gamma^*_{\text{S-adap}}$*-admissible environments and protocol* Π_{CSN} *with parameter* $s > k + \frac{2(1+\epsilon)}{\mathsf{q}_{\text{M-adap}} \cdot \epsilon^2 \log_2 e} h$, $\frac{k}{(1-2\rho)\omega} + 2k < L < \frac{\tau_{\text{obs}}(1-\epsilon)\alpha}{2}$, *and* $d \geq \max(\frac{k}{-\log \mathsf{q}_{\text{M-adap}}}, 4 \log s)$*. Given that the bootstrapping condition is satisfied. Then, protocol* Π_{CSN} *achieves persistence and liveness; and sparsity.*

2.4.3 S-adaptive $\langle \phi, \gamma \rangle$ security

We consider *mining power concentration*[4], in which the top γ fraction of honest nodes control at least a fraction γ of the mining power, for some $\gamma > \rho$. With mining power concentrated in the top few nodes, it is possible to reduce the number of core nodes, thus reducing network sparsity. More concretely, since mining power is concentrated in a small fraction of nodes, the CoSpaN protocol requires a smaller core width so that the nodes selected as core control a large fraction of the mining power. The adversary can not control more than a fraction ρ of mining powers, we can guarantee that a big enough number of core nodes are honest.

Consider admissible environments with a predicate Γ^*_{conc} in which the honest majority assumption holds. Now we say it holds that $\Gamma^*_{\text{conc}}(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}, \phi, \gamma) = 1$ if for all $n, Q, \delta, \tau_{\text{obs}} \in \mathbb{N}$, $\eta, \rho, \phi, \gamma \in (0, 1)$, there exists $k = \Omega(\kappa)$ such that

$$\Gamma^*_{\text{S-adap}}(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}) = 1 \quad \text{and} \quad \gamma > \rho.$$

Definition 29 (Γ_{conc}-admissible environments)**.** *We say a tuple* $(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}, \phi, \gamma, \mathcal{A}, \mathcal{Z})$ *with* $\Gamma_{\text{conc}}(n, Q, \eta, \rho, \delta, \tau_{\text{obs}}, \phi, \gamma) = 1$ *is* Γ_{conc}*-admissible w.r.t* $(\Pi^{\mathsf{V}}, \Lambda)$ *if* $\mathcal{A}$*,* $\mathcal{Z}$ *are probabilistic polynomial-time algorithms, and for every* VIEW *in the support of* $\mathsf{EXEC}_{\Pi^{\mathsf{V}}, \mathcal{A}, \mathcal{Z}}(\kappa)$*, the following holds:*

1. *Conditions (1), (2), (3), (4) in Definition 24.*

[4]For example, in Bitcoin [1], more than 50% of the mining power is controlled by the top 7 mining pools.

2. The top ϕ fraction honest nodes control at least a fraction γ of mining power

As the mining power is concentrated in a small fraction of nodes, the nodes that are selected as core control a large fraction of mining power. We set the core width such that top ϕ fraction of nodes are selected as core nodes. In other words, the set of core nodes control at least a fraction γ of mining power. Since the adversary can not control more than a fraction ρ of mining powers, it cannot corrupt all core nodes. Thus, we can show that the number of honest core nodes is still big enough.

Lemma 30 (Honest core nodes in S-adaptive setting with mining power concentration)**.** *For any $\epsilon > 0$, let $\mathsf{q}_{\text{conc}} = (\gamma - \rho)\omega - \frac{k}{L_{\text{reg}}} - \epsilon$. Consider protocol Π with parameter $s > k + \frac{2(1+\epsilon)}{\mathsf{q}_{\text{conc}} \cdot \epsilon^2 \log_2 e} h\phi$, we have,*

$$\Pr[h_{\text{c}} \geq s \cdot \mathsf{q}_{\text{conc}} \mid \mathcal{S}_i \wedge \mathcal{R}_i] > 1 - e^{-\Omega(\kappa)}.$$

Proof. Let K be the set of the top honest nodes that control at least a fraction γ of honest mining power. Consider a set $W \subseteq K$ in which the total mining power of all nodes in W is at least $\gamma - \rho$.

Let r_1 be the first round in the execution in which the length of the chain of any honest node is at least iL, i.e., all honest nodes are at epoch i.

For any $\epsilon_1 > 0$, let $r_2 = r_1 + (1 - \epsilon_1)\frac{L_{\text{reg}} - k}{Q \cdot p}$. From round $r_1 + 1$ to round r_2, nodes make at most $(1 - \epsilon_1)\frac{L_{\text{reg}} - k}{p}$ queries to random oracle in which the probability of success in generating a new block of a single query is p. We represent each query by a Bernoulli random variable with an expected value of p. Let n_{b} be the number of blocks that are generated from round $r_1 + 1$ to round r_2. By using the Chernoff bound in Lemma 56 on at most $(1 - \epsilon_1)\frac{L_{\text{reg}} - k}{Q \cdot p}$ Bernoulli random variables, we have,

$$\Pr[n_{\text{b}} < k] 1 - e^{-\Omega(\kappa)}.$$

Using the Chernoff bound in Lemma 56, we have,

$$\Pr[n_{\mathrm{b}} < (1-\epsilon_1)(1+\epsilon_1)(L_{\mathtt{reg}} - k)]1 - e^{-\Omega(\kappa)},$$
$$\Rightarrow \Pr[n_{\mathrm{b}} < (L_{\mathtt{reg}} - k)]1 - e^{-\Omega(\kappa)}.$$

Let $\bar{h_{\mathrm{c}}}$ be the number of registration blocks that are generated by the honest nodes that are willing to participate as core nodes from round r_1 to r_2. In each round, the honest nodes in W make at least $(\gamma - \rho)\omega$ queries to the random oracle. Thus, from round r_1 to round r_2, the honest nodes in W make at least $(1-\epsilon_1)\frac{L_{\mathtt{reg}}-k}{p}(\gamma-\rho)\omega$ queries to the random oracle. Here, the probability of success in generating a new registration block of a single query is $p' = p\frac{s}{L_{\mathtt{reg}}}$. We represent each query by a Bernoulli random variable with an expected value of p'. We choose $\epsilon_2 > 0$ such that $1 - \epsilon = (1-\epsilon_1)(1-\epsilon_2)$. Using the Chernoff bound in Lemma 56, we have,

$$\Pr[\bar{h_{\mathrm{c}}} > (1-\epsilon)(\gamma-\rho)(1-\frac{1}{L_{\mathtt{reg}}})\omega \cdot s] > 1 - e^{-\Omega(\kappa)}.$$

At round r_2, with a probability of 1 - $e^{-\Omega(\kappa)}$, we have, 1) the length of the longest chain is at most $iL + L_{\mathtt{reg}} - k$; and 2) the number of registration blocks that are generated by honest nodes in W who are willing to participate as core nodes is at least $(1-\epsilon)(\gamma-\rho)(1-\frac{1}{L_{\mathtt{reg}}})\omega$.

From chain quality property in Lemma 12, with probability $1 - e^{-\Omega(\kappa)}$, there exist an honest block from the $(iL + L_{\mathtt{reg}} - \kappa + 1)$-th blocks to the $(iL + L_{\mathtt{reg}})$-th blocks. Thus, if an honest node register as a core node (by propagating a registration block) before the $(L_{\mathtt{reg}} - k)$-th block of the registration period is generated, it will be selected as a core node in the next epoch (i.e., the registration block is included on

the blockchain). Thus, we have,

$$\Pr[h_{\mathrm{c}} \geq \bar{h_{\mathrm{c}}}] > 1 - e^{-\Omega(\kappa)},$$
$$\Rightarrow \Pr[h_{\mathrm{c}} \geq \mathsf{q}_{\mathrm{conc}} \cdot s] > 1 - e^{-\Omega(\kappa)}.$$

Following the same proofs and Theorem 28 (except replacing $\mathsf{q}_{\text{M-adap}}$ by $\mathsf{q}_{\mathrm{conc}}$), we can prove the security of the protocol Π_{CSN} in the presence of an adaptive adversary.

□

Following the same proofs and Theorem 28 (except replacing the core quality $\mathsf{q}_{\text{M-adap}}$ by $\mathsf{q}_{\mathrm{conc}}$), we can prove the security of the protocol Π_{CSN} in the presence of an adaptive adversary.

Theorem 31 (S-adaptive $\langle \phi, \gamma \rangle$)**.** *Consider $\Gamma^{*}_{\mathrm{conc}}$-admissible environments and the protocol Π_{CSN} with parameters $\frac{k}{(\gamma-\rho)\omega} + 2k < L < \frac{\tau_{\mathrm{obs}}(1-\epsilon)\alpha}{2}$, $s > k + h\phi$ and $d \geq \max(\frac{k}{-\log \mathsf{q}_{\mathrm{conc}}}, 4 \log s)$. Under the bootstrapping condition is satisfied, protocol Π_{CSN} achieves persistence and liveness; and sparsity.*

2.5 Numerical Studies

We compare the CoSpaN protocol to existing Bitcoin-like protocols by analyzing the costs for adversaries to perform various attacks. Additionally, we analyze the defense costs in the proposed threat models and the effect of mining power concentration on reducing defense costs.

2.5.1 Setup

Protocols. We consider CoSpaN and the following two protocols.

- Bitcoin-L(ike):This protocol mimics the current protocol for the Bitcoin P2P network. Each node makes $d = 8$ outbound connections and accepts up to $d_{max} = 125 \approx 15d$ inbound connections. Nodes follow the proposed defenses in

[56] to randomize their connections, i.e., 1) each node makes connections to $d = 8$ random nodes, and 2) if a node receives more than d_{max} requests for inbound connections, it randomly selects d_{max} nodes to accept the connections. Note that the adversary can no longer perform preferential attachment attacks to increase the probability of honest nodes establishing outbound connections toward their controlled nodes. However, we assume that each adversary-controlled node will make connections to *all* honest nodes to get more slots from the 125 inbound connections to honest nodes.

– Bitcoin-R(andom): This protocol considers a more advanced topology construction, building a truly random topology over the participating nodes. Any two nodes have exactly the same probability $\frac{d}{n}$ of having a connection. Thus, the expected number of connections for each node is d, and there is no limit on the number of inbound connections. In this protocol, the adversary can no longer get more slots within the 125 connection limit of the honest nodes. However, the adversary can increase the number of connections to honest nodes by creating more sybil nodes (higher η).

Note that with the same connection parameter d, all three protocols will construct networks of similar sparsity (average degree).

Parameter settings. We consider a network with $1,000$ honest nodes, $1,000$ malicious nodes, and additional $2,000 \cdot n_s$ Sybil nodes (of zero-mining power). Thus, the fraction of malicious node $\eta = \frac{1}{2(1+n_s)}$. Without otherwise mentioned, we consider a static adversary that controls a fraction $\rho = 0.3$ of mining power, the participation rate in CoSpaN protocol is $\omega = 0.9$. For CoSpaN protocol, we set the core width to be 5% the number of nodes in the network.

Metrics. We measure the cost for the adversary to break the security of the protocols

including 1) the *Sybil factor* needed for the adversary to disconnect the network (thus, violating the reliable dissemination requirement) and 2) the *adversarial mining power* needed for the adversary to perform a double spending attack. We also study the required *honest mining power* to guarantee the security of CoSpaN protocol in different settings and analyze the *degree distribution* and *its correlation to mining power distribution*. All the experiments are repeated $1,000$ times and the average numbers were reported.

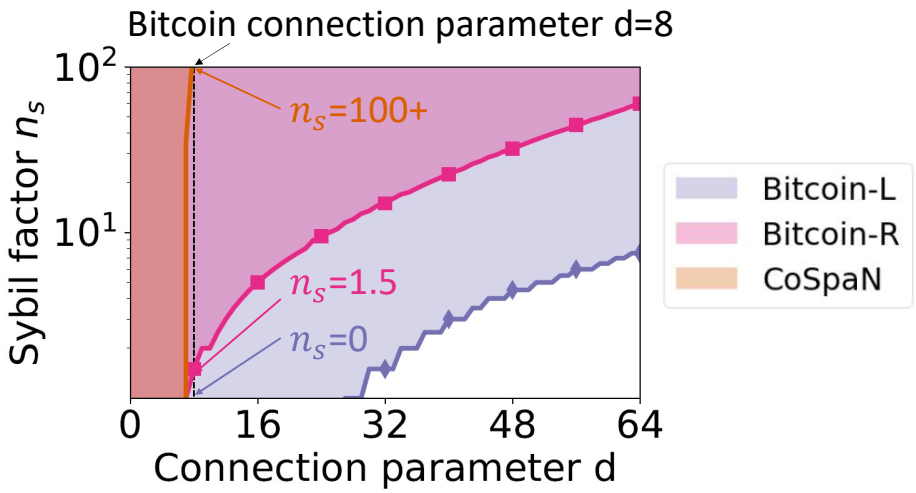


Figure 8: Protocols' resiliency against Sybil attacks. For each network connection parameter (d), the shaded areas above the curves show the corresponding Sybil factors that the adversary needs to break the protocols' security. The higher Sybil factor that a protocol can withstand, the stronger security. At the Bitcoin's connection parameter $d = 8$ (the dashed line), the adversary can disrupt Bitcoin-L's network without any Sybil nodes ($n_s = 0$), disrupt Bitcoin-R's network with $n_s \approx 1.5$, but can only disrupt CoSpaN's network for unrealistically high $n_s > 100$.

2.5.2 Costs of attacks

We compare the protocols by showing the cost for the adversary to perform certain attacks in static settings.

2.5.2.1 Sybil attacks

We consider a static adversary that controls 30% of the mining power. The adversary attempts to disrupt the network by creating more Sybil nodes, i.e., increasing the Sybil factor, in order to break reliable dissemination. Figure 8 shows that the cost to break the reliable dissemination of the CoSpaN protocol is significantly higher than that of the other protocols. For example, at Bitcoin's sparsity level with $d = 8$, the adversary needs a Sybil factor $n_s > 100$ to break the reliable dissemination of the CoSpaN protocol, while the numbers for Bitcoin-L and Bitcoin-R are only $n_s = 0$ and $n_s = 1.5$, respectively. Even at a much higher connection parameter ($d = 64$), the adversary can still break the reliable dissemination security requirement of Bitcoin-L and Bitcoin-R with a high Sybil factor ($n_s < 100$).

2.5.2.2 Double-spending attacks

We consider a static adversary with a Sybil factor of $n_s = 10$ and a connection parameter of $d = 8$. The adversary performs a double-spending attack by 1) splitting the network, and thus the honest mining power, and 2) using the block withholding strategy [88]. In Figure 9, we show the required mining power for the adversary to successfully reverse a 12-confirmation transaction with a probability of more than 10%. For the CoSpaN protocol, the adversary always needs 51% of the mining power to perform the attacks, as it cannot split the network. For the Bitcoin-L and Bitcoin-R protocols, the honest nodes are disrupted into multiple components. Hence, the

adversary can perform the double-spending attack in each component with much less mining power. For example, if the Sybil factor is $n_s = 100$ and the connection parameter is $d = 8$, the adversary only needs 4% of the mining power to attack the Bitcoin-L network and 20% of the mining power to attack the Bitcoin-R network.

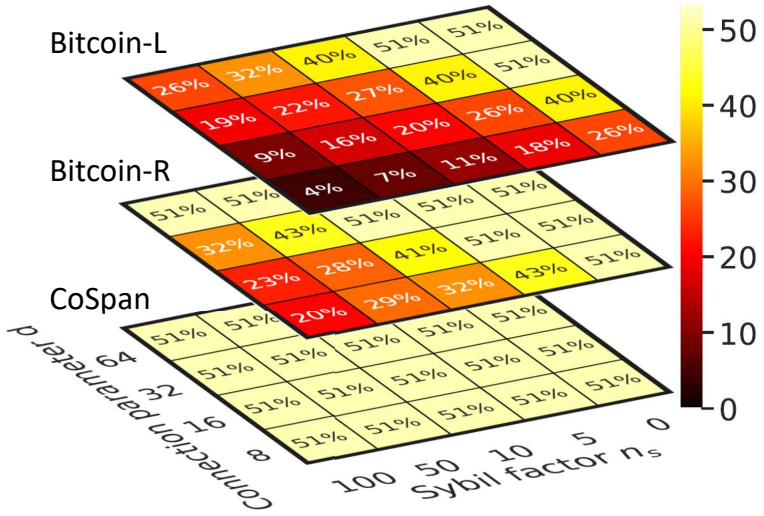


Figure 9: Double-spending attacks with less than 51%. The required mining power by an adversary to reverse a 12-confirmation transaction with a more than 10% probability in each protocol. The adversary is able to split the networks in Bitcoin-L and Bitcoin-R (but not CoSpaN) protocols, thus, only needs to exceed the mining power of the largest connected component.

2.5.3 CoSpaN in different security settings

We will now analyze the security of the CoSpaN protocol in different settings. We will omit Bitcoin-L and Bitcoin-R as they cannot achieve the desired security properties in (weakly) adaptive settings. Figure 10 shows the required honest mining power to achieve security in the network layer of the CoSpaN protocol with different parameter settings (connection parameter and core width). We will consider four security settings: static, M-adaptive, S-adaptive (with uniform mining power), and

S-adaptive $\langle 0.1, 0.5 \rangle$.

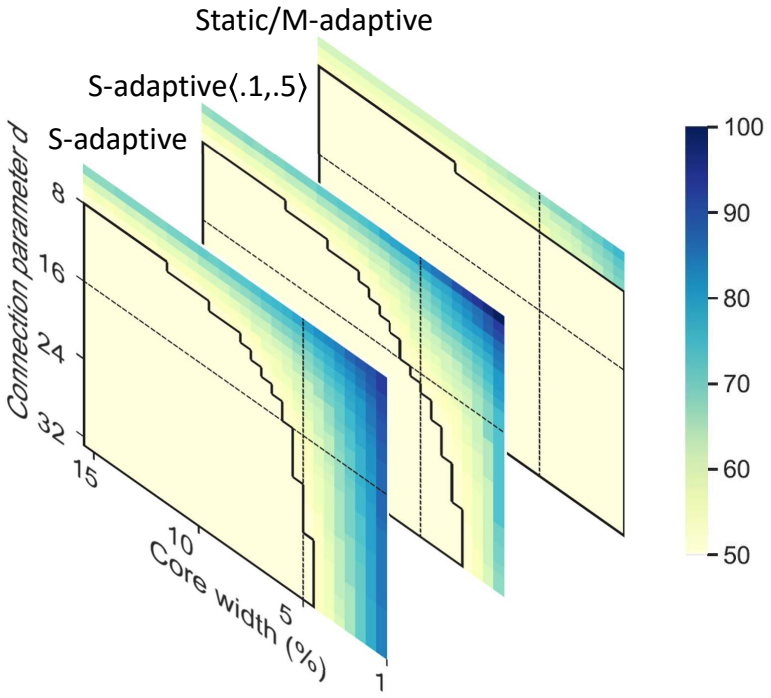
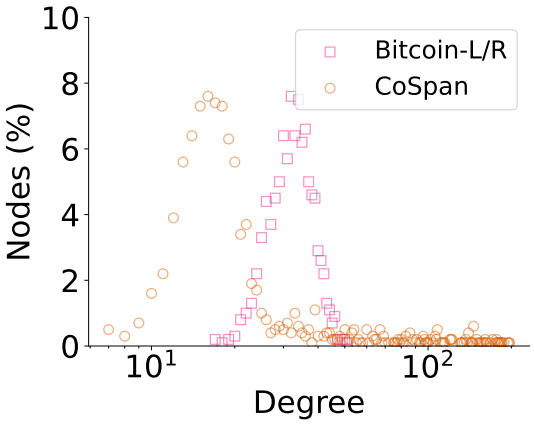
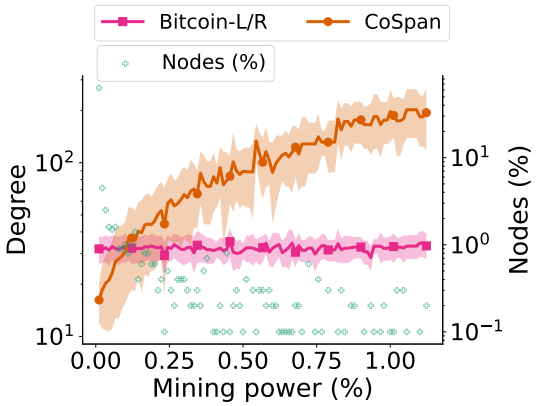


Figure 10: The required honest mining power to achieve reliable dissemination for CoSpaN protocol. The area under the curve show when the same requirement (51%) in PSS is met.

In the S-adaptive setting, there is a trade-off between the connection parameter and the core width. When the core width increases, it requires a smaller connection parameter, and vice versa. For example, when the core width increases from 5% to 15% to achieve the security of 51% of honest mining power, the connection parameter d decreases from 22 to 8. Mining power concentration helps to reduce the required core width. For example, with a core width of 5%, the required connection parameter in an S-adaptive setting with no mining power concentration (i.e., all nodes have the same mining power) is 22. Meanwhile, the required connection parameter in S-adaptive $\langle 0.1, 0.5 \rangle$ is 17.



(a) Degree distribution distribution.



(b) The correlation between mining power and degree of nodes. The shaded areas show the 99% confidence intervals.

Figure 11: The correlation between the mining power and the degree of nodes in CoSpaN and Bitcoin-random network. The distribution of Bitcoin-L and Bitcoin-R are the same.

2.5.4 Network characteristics

Consider a network in which the mining power of nodes follows a power law distribution. We analyze the degree distribution and the relation between the degree distribution and the mining power distribution of the honest nodes. From Figure 11a, the degree distribution of Bitcoin-L and Bitcoin-R networks follows a normal distribution. The degree distribution of the CoSpaN network follows a power law distribution. Specifically, as shown in Figure 11, the degrees of nodes in Bitcoin-L and Bitcoin-R networks are independent of the mining power. The degree of nodes in those networks has the same distribution. Meanwhile, in the CoSpaN network, the degree of nodes is proportional to the mining power.

2.6 Conclusion

This paper presents CoSpaN, the first network protocol designed for sparse networks in the PoW setting. Nodes use merge-mining to provide proofs of their mining power, and establish verifiable random connections such that the expected number of connections for each node is proportional to its mining power. Additionally, the CoSpaN protocol preserves the confidentiality of core nodes, meaning that the adversary cannot determine whether or not a node is associated with a core node. As a result, the adversary is unable to launch DoS attacks to bring down core nodes and disrupt the network. Plus, each connection is known only by the two participating nodes. This is a key feature for achieving reliable dissemination in a sparse network against an adaptive adversary.

Our CoSpaN protocol can be utilized to build a backbone network for the dissemination of critical information such as block headers. Additional connections can be added to increase the throughput and performance of the blockchain systems. A future direction is to investigate whether or not we can construct a secure consensus protocol with nodes of bounded degree.

2.7 Supplemental materials

2.7.1 Nakamoto's protocol $\Pi_{\mathbf{Nak}}$

In Nakamoto's protocol Π_{Nak} [89], each node maintains a chain, i.e., a sequence of blocks linked by hash values. Each block is a tuple $(\mathsf{context}, txs, nonce)$, where txs the list of transactions that the node received from the environment, the $\mathsf{context}$ consists of a pointer (hash value) prev to the previous block, the Merkle root $MkRoot(txs)$, a single hash value to prove the validate the included transactions following the approach in the Bitcoin's header.

Also, we add to the context a *network pseudo-identity* cid, that is needed in core selection,

$$\mathsf{context} = \langle \mathsf{prev}, MkRoot(txs), \mathsf{cid} \rangle. \tag{2.8}$$

The algorithm Λ obtains the ledger from a chain $\mathcal{C}$ by truncating the last k blocks for some integer $k = O(\kappa)$ and returns the list of transactions [88]. The nodes compete to become *block producers* via a PoW mechanism. The block producer will generate and propagate a new block to all the nodes.

Block producer selection. In each round, nodes makes attempts to solve a PoW puzzle by searching for a *nonce* over a $\mathsf{context}$ that satisfies the hash inequality

$$H(\mathsf{context}, nonce) < \mathtt{T},$$

where $\mathtt{T}$ is the *mining difficulty*.

If the hash inequality is satisfied, the node becomes a block producer, adds a new block $(\mathsf{context}, txs, nonce)$ to the longest (best) chain, and propagate the new chain to all the neighbors (who will forward the chain further). Before adding a new chain $\mathcal{C}$ to the local state, each node verifies the validity of 1) the hash inequalities in all blocks and 2) the list of transactions in the chain, using a predicate V.

2.7.2 Verifiable Random Functions

In our design, a verifiable random function (VRF) has been used for core miners to establish connections with other miners. Here, we present formal definitions of Verifiable Random Functions in [40].

Definition 32. *A function family* $F_{(\cdot)}(\cdot) : \{0,1\}^{\mathsf{a}(\kappa)} \to \{0,1\}^{\mathsf{b}(\kappa)}$ *(where* $\mathsf{a}(\cdot), \mathsf{b}(\cdot)$ *are polynomials) is a family of VRFs if there exist algorithms* $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ *such that:*

- *The algorithm* Gen *takes as input a security parameter* 1^κ *and outputs a key pair* $(\mathsf{sk}', \mathsf{pk}')$.
- *The algorithm* Prove *takes as input a private key* sk'*, a string* x*, and outputs a pair* (σ, π)*, where* $\sigma = F_{\mathsf{sk}'}(x)$.
- *The algorithm* Verify *takes as input a public key* pk'*, a string* x*, an output* σ*, a proof* π *and verifies that* $\sigma = F_{\mathsf{sk}'}(x)$ *using the proof* π*. It output* 1 *if* y *is valid and* 0 *otherwise.*

Additionally, we require the following properties:

1. ***Uniqueness.*** *No values* $(\mathsf{pk}', x, \sigma, \sigma', \pi, \pi')$ *can satisfy both*

$$\mathsf{Verify}_{\mathsf{pk}'}(x, \sigma, \pi) = 1 \text{ and } \mathsf{Verify}_{\mathsf{pk}'}(x, \sigma', \pi') = 1$$

   unless $\sigma = \sigma'$

2. ***Provability.*** *If* $(\sigma, \pi) := \mathsf{Prove}_{\mathsf{sk}'}(x)$*, then* $\mathsf{Verify}_{\mathsf{pk}'}(x, \sigma, \pi) = 1$.

3. ***Pseudorandomness.*** *For all* PPT *adversary* $\mathcal{A} = (\mathcal{A}_E, \mathcal{A}_J)$ *that does not query the oracle on* x*, we have*

$$\Pr\left[\begin{array}{l|l} (\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Gen}(1^\kappa); & \\ (x, st) \leftarrow \mathcal{A}_E^{\mathsf{Prove}_{\mathsf{sk}'}(\cdot)}; & \\ y_0 := F_{\mathsf{sk}'}(x); y_1 \leftarrow \{0,1\}^{\mathsf{b}(\kappa)}; & b = b' \\ b \leftarrow \{0,1\}; & \\ b' \leftarrow \mathcal{A}_J^{\mathsf{Prove}_{\mathsf{sk}'}(\cdot)}(y_b, st) & \end{array}\right] \leq \frac{1}{2} + \varepsilon(\kappa).$$

2.7.3 Chernoff bound

We provide here the Chernoff bound for Bernoulli random variables [52] that we use in the proofs.

Lemma 33. *Suppose* $\{Z_i : i \in [t]\}$ *are independent and identically distributed Bernoulli random variables with* $\Pr[Z_i = 1] = \mu$*, for all* $i \in [t]$*. Then, for any* $\epsilon > 0$*, we have*

$$\Pr[\sum_{i=1}^{t} Z_i \leq (1-\epsilon)t\mu] \leq e^{-\epsilon^2 t\mu/2},$$
$$\Pr[\sum_{i=1}^{t} Z_i \geq (1+\epsilon)t\mu] \leq e^{-\epsilon^2 t\mu/3}.$$

CHAPTER 3

THE NETWORK LAYER: FAST SYNCHRONIZATION

In this chapter, we examine the impact of heterogeneity in network bandwidth on system synchrony, throughput, and latency, i.e., how the system security and performance are affected by the weakest nodes in the P2P network. Our findings reveal that certain intelligent (centralized) designs of the P2P network can counteract the lack of bandwidths of the weakest nodes, achieving near-optimal throughput and latency. We identify the key principles of our centralized design, which we use to develop a practical decentralized approach for P2P networks called ProSHeN. ProSHeN incorporates half-duplexity, constant-capacity channel, and early burst to significantly enhances the network synchrony. In our comprehensive evaluation, ProSHeN shows substantial improvement in both security and performance, enhancing the blockchain resiliency against block withholding and selfish-mining attacks.

We present a model of networks with heterogeneity bandwidth in Section 3.1 and investigate the theoretical limit of blockchain network in Section 3.2. In Section 3.3 we present the propagation schemes in heterogeneous networks. We show the analysis of the propagation schemes in Section 3.4. The experiments are shown in Section 3.5.

3.1 Model

Abstracting a blockchain system. Consider a blockchain system with n participants that are modeled as a set of nodes $V = \{1, 2, \ldots, n\}$. Each node $i \in V$ has

a upload bandwidth[1] c_i. The set of nodes adhere to a consensus protocol to communicate with each other and maintain a blockchain. The blockchain consists of a sequence of blocks that contain transactional data. The mining power of each node determines its ability to add new blocks to the chain, and nodes with higher mining power can generate and contribute more blocks to the blockchain.

Data arrival rate. During the execution of the blockchain system, some source node i may produce new data (e.g., blocks, transactions) and propagate the data to other nodes. We denote the *data arrival rate* λ_i as the average amount of data that is produced by node i. Normally, the total data arrival rate of all nodes is considered the throughput of the blockchain system. However, this may not hold true when the data arrival rate is too high (e.g., when the system is under DoS attacks). When the total data arrival rate exceeds the network's capacity, the network becomes highly asynchronous, and nodes may receive different data. In the best-case scenario, nodes can achieve consensus on the data received by all nodes.

DoS attacks. The adversary can increase the data arrival rate by performing a DoS attack, such as a transaction flooding attack. For instance, the Solana network, which has a multi-billion dollar market cap, experienced a few hours of downtime due to flooding attacks[2]. In this type of attack, the adversary creates a large number of transactions, including empty ones, which act as congestion material and overload the network, causing it to go offline.

Our goal is to answers the following research questions:

---

[1]Today in the Internet, the download bandwidth is much larger than the upload bandwith, e.g., the case of ADSL. Thus, we assume, for simplicity, that all nodes have sufficient download bandwidths so that time to transmit data between two nodes depend on the upload bandwidth of the sender but not the download bandwidth of the receivers.

[2]https://u.today/solana-network-goes-offline-again-now-ddos-attack-may-be-reason

1. *What level of DoS attack can a heterogeneous network handle?*

2. *Is there a way to make the heterogeneous network handle DoS attacks?*

3.2 Theoretical limit of the blockchain network

To answer the first question on the level of DoS attack that a heterogeneous network can handle, we study the theoretical limit of the blockchain network. For simplicity, we consider that data can be divided into packets of arbitrarily small sizes. Further, different packets at an intermediate node can be mixed together using network coding [4, 59] before forwarding. This will help nodes to receive only novel information from peers [4, 59].

Transmission schedule. For a source node $i \in V$ a target node $j \in V$, we characterize the data flow from i to j. We denote the data on an edge $(u, v) \in E$ of the flow from i to j by a non-negative variable $f_{uv}^{(i,j)}$.

Flow's constraint. For each node $u \in V$, the flow's constraint from i to j at u gives us

$$\sum_{v \in V} f_{vu}^{(i,j)} - \sum_{v \in V} f_{uv}^{(i,j)} + -\lambda_i \mathbb{1}_{\{u=i\}} - \lambda_i \mathbb{1}_{\{u=j\}} = 0, \forall u \in V, \forall i, j \in V. \tag{3.1}$$

Capacity's constraint. Also, we denote $f_{uv}^{(i)}$ is the data originated from s flowing through the edge (u, v), i.e.,

$$f_{uv}^{(i)} = \max_{j \in V \setminus \{i\}} f_{uv}^{(i,j)}, \forall i \in V, u, v \in V. \tag{3.2}$$

and g_{uv} is the total data transmission on the edge (u, v), i.e.,

$$g_{uv} = \sum_{s \in V} f_{uv}^{(i)}, \forall s \in V, (u, v) \in E. \tag{3.3}$$

The upload capacity at node u is capped by c_u, thus, the capacity's constraint at node u give us

$$\sum_{v \in V} g_{uv} \leq c_u \ \forall \ u \in V. \tag{3.4}$$

Feasible transmission schedule. Consider an *arrival rate* $\lambda = (\lambda_1, \cdots, \lambda_n)$ in which each node $i \in V$ produce new data with the rate of λ_i. We say a transmission schedule f is a *feasible transmission schedule* if it satisfies the flow's constraint in Eq.3.1 and the capacity's constraint in Eq.3.4.

Throughput. Given a set of nodes $V = (1, \cdots, n)$ and the upload capacity $C = (c_1, \cdots, c_n)$, we say an algorithm can achieve a throughput of TP if for all arrival rate $\lambda = (\lambda_1, \cdots, \lambda_n)$ such that $\sum_{i \in V} \lambda_i \leq \mathsf{TP}$, the algorithm can always returns a feasible schedule f of λ.

Throughput optimization. In [69], Kumar and Ross showed, in Theorem 1, that the maximum throughput is upper bounded by

$$\min \left\{ d_{min}, \frac{\sum_{i=1}^{n} c_i}{n-1} \right\},$$

where d_{min} is the minimum download bandwidth of any receiver.

Under our assumption, the upper-bound on the optimal throughput can be simplified into

$$\mathsf{OPT}_{TP} \leq \frac{\sum_{i=1}^{n} c_i}{n-1}. \tag{3.5}$$

While the bound is derived using a fluid model [69], it remains true in general. More importantly, the optimal throughput is realizable under the fluid model by through a feasible transmission schedule in [69]. Using the same transmission schedule, we can also prove that the optimal throughput is achievable if network coding is deployed [59]. The transmission schedule is, however, not scalable as it requires each

node to connect and forward the data to all other nodes, resulting in an $O(n^2)$ wiring scheme with fragmented data.

Latency. The latency is the time for any source nodes to transmit the data to all other nodes. Almost all existing studies on optimal throughput [69, 97, 4, 59, 75] assume a synchronous setting in which data transmission incurs zero-delay. As a consequence, the minimum propagation time $\frac{|B|}{\mathsf{OPT}_{TP}}$ of a block B (with block size $|B|$) is achieved when the throughput is maximized at OPT_{TP}.

We propose a model that incorporate transmission delay at each link. Consider a source node s, for each target node t, the delay $\delta(p)$ incurs by a packet along a path p from s to t is defined as the total latencies of all edges in p, i.e., $\delta(p) = \sum_{(i,j)\in p} \delta_{ij}$. Let $f_p = \min_{(u,v)\in p} f_{u,v}^{(i,j)}$ be the amount of data travel through path p from s to t.

Distribution time. Given a transmission schedule (f, g), the time to deliver block B from s to t will be

$$\mu(s,t) = \inf_{T>0} \left\{ \sum_{p\in P_{s,t}} f_p \max\{0, T - \delta(p)\} \geq |B| \right\}, \tag{3.6}$$

where $P_{s,t}$ is the set of all paths from s to t.

For each path p, if the source start transmission along p at time 0, then the source will start receiving data at time $\delta(p)$. Thus, the amount of data received through p at time T will be $f_p \max\{0, T - \delta(p)\}$. Observe that only the paths p with $\delta(p) < \mu(s,t)$ help in delivering the data from s to t.

Latency. The latency is the maximum distribution time from any source node s to any target node t, i.e.,

$$\mu = \max_{s,t\in V} \mu(s,t) \tag{3.7}$$

Network sparsity. In a transmission schedule, we allow all pairs of nodes to prop-

agate data to each other. However, for most of the pairs, the total data transmission should be 0. Given a transmission schedule f, We define an (directed) overlay network $G_f = (V, E_f)$ for a transmission schedule by including all of pairs of nodes that have non-zero data transmission links in E, i.e.,

$$E_f = \{(i, j) : i, j \in V, g_{i,j} > 0\} \tag{3.8}$$

The network sparsity is defined over the number of links in E_f. We say a network is sparse if the number of link in E_f is $O(dn)$ for some parameter $d > 1$.

Definition 34 (Blockchain P2P Network Design Problem (BlockNetDP))**.** *Consider a P2P network consisting of n nodes $\{1, 2, \ldots, n\}$ in which node i has upload capacity $C(i)$ (and unbounded download capacity), and transmision delay d_{ij} between i and j, for all $i, j \in [n]$. Let $\lambda_i \geq 0$ be the rate that node i broadcast data to all network for $i \in [n]$. Design a P2P network that admit a feasible transmission schedule with rate $\lambda = \{\lambda_i\}$ subjecting to the following constraints*

- *Capacity constraint: $\forall i \in V : \sum_{j \in V} g_{ij} \leq C(i)$*
- *Sparsity constraint: The number of links is at most $O(dn)$, for some constant $d > 1$.*

At the same time, we aim to minimize the propagation time*, i.e., the time for any source nodes to propagate the data to all other nodes.*

3.3 Propagation Scheme in Heterogeneous Networks

3.3.1 ProSHeN: A Propagation Scheme in Heterogeneous Networks

We present ProSHeN, a **Pro**pagation **S**cheme in **He**terogeneous **N**etworks. The transmission schedule can achieve near-optimal throughput. Plus, the data will be received by all nodes with $O(\log n)$ hops.

3.3.1.1 Overview

The main idea of our algorithm is building d *directed broadcast trees* or *broadcast trees* with disjoint links, in which all links in the trees have the same capacity. We denote by w the capacity in each link. The data will be distributed to all nodes through the directed broadcast trees.

Algorithm 1: ProSHeN algorithm

Input : Given the set of node $V = (1, \cdots, n)$, the upload capacity $C = (c_1, \cdots, c_n)$, the arrival rate $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$, the number of broadcast tree d, and the capacity of each tree w.

Output: Return the transmission schedule f.

1 $S = \mathsf{LinkAllocation}(C, \lambda, d, w)$

2 $T = \mathsf{BuildBroadcastTrees}(S)$

3 $f = \mathsf{ConstructSchedule}(\lambda, T, w)$

4 Return f

The ProSHeN algorithm consists of three steps (see Algorithm 1) as follows. First, based on the capacity of nodes, we *assign links into trees* such that we are able to build the trees with $O(\log n)$ depth. Then, we *build broadcast trees* in which, in each tree, the nodes that assigns more links is closer to the root nodes (which is the node that assigns the most links). Finally, to *construct the transmission schedule*, the source first forward the arrival data to the root nodes. Then, the data will be forwarded through the links of the broadcast trees. In Figure 12, we illustrate an example of constructing multiple broadcast trees for a set V of 7 nodes. In Figure 13, we show how nodes forward data from the source node 1 through the red broadcast tree.

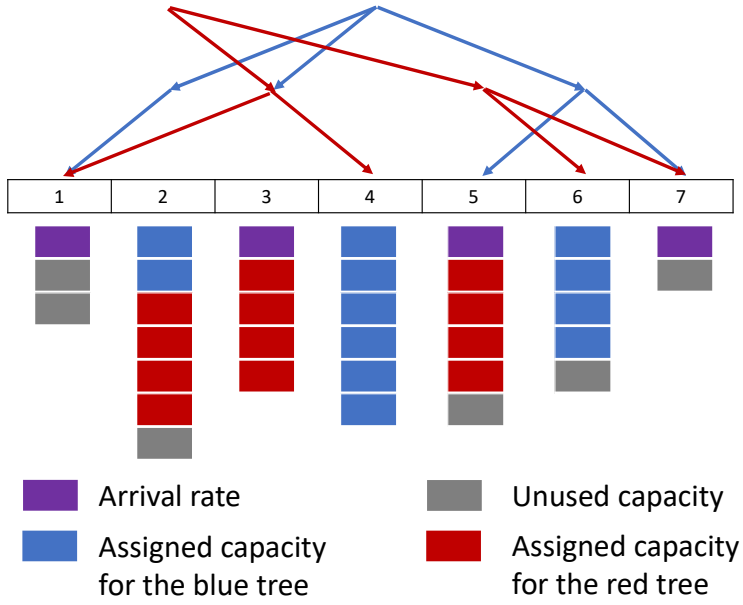


Figure 12: Constructing 2 broadcast trees with the capacity $w = 2$ for the set V of 7 nodes, the capacity $C = (3, 7, 5, 6, 6, 5, 2)$, the arrival rate $\lambda = (1, 0, 1, 0, 1, 0, 1)$. Nodes assign $(0, 1, 0, 3, 0, 2, 0)$ links to the blue tree and $(0, 2, , 2, 0, 2, 0, 0)$ links to the red tree.

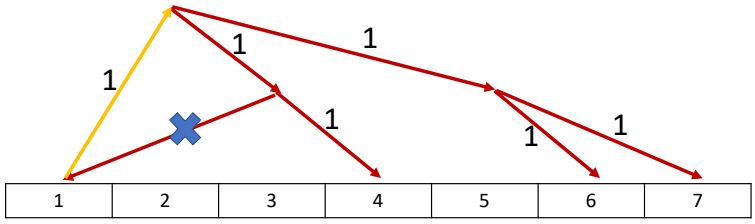


Figure 13: Forwarding data from the source node 1 through the red broadcast tree. As node 1 already had the data, we do not use the link $(3, 1)$ to forward data.

3.3.1.2 Assignment of Links to Trees

Consider a set of node $V = (1, \cdots, n)$, an upload capacity $C = (c_1, \cdots, c_n)$, and an arrival rate $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$. We first assign the links of each nodes into d broadcast trees in which each trees consist of $n-1$ links with capacity w. Furthermore, the links should be assigned in the way that we are able to construct the trees with $O(\log n)$ depth.

Algorithm 2: AssignTrees algorithm.

Input : Given the upload capacity C, the arrival rate λ and the parameters d, w.

Output: Return the set of links trees S.

$L = \left\{\ell_i = \lfloor \frac{c_i - \lambda_i}{L} \rfloor\right\}_{i\in[n]}$, $L' = \{\ell'_i = l_i\})_{i\in[n]}$

$q_k = n-1, \forall k \in [d]$

$S = \{s_{k,i} = 0\}_{k\in[d], i\in[n]}$

while $\exists q_k > 0$ *and* $\exists \ell'_i > 0$ **do**

  $k = \arg\max_{k\in[d]} q_k$, $i = \arg\max_{i\in[n]} \ell'_i$

  $nl = \min(\ell'_i, q_k, \lfloor \frac{n}{4} \rfloor)$

  $s_{k,i} = s_{k,i} + nl$, $q_k = q_k - nl$, $\ell'_i = \ell'_i - nl$

Return S

Let $S = \{s_{k,i}\}_{k\in[d], i\in V}$ be the assigned matrix, where $s_{k,i}$ is the number of assigned links of node i in the k-th tree. The number of assigned links in each tree equals $n-1$, i.e.,

$$\sum_{i\in[1,n]} s_{k,i} = n-1, \forall k \in [d] \tag{3.9}$$

Additionally, to construct the tree with $O(\log n)$ depth, we limit the number of *single-link nodes* in each tree. Here, we say node i is a single-link node of the k-th

tree if $s_{k,i} = 1$. To be precise, the number of single-link nodes in each tree should be at most $\frac{3n}{4}$, i.e.,

$$\sum_{i \in [1,n]: s_{k,i}=1} s_{k,i} \leq \frac{3n}{4}, \forall k \in [d] \tag{3.10}$$

Let ℓ_i be the maximum number of links that node i can contribute in all trees. Since each node i need to reserve λ_i bandwidth to send the data to the root nodes, we have,

$$\ell_i = \lfloor \frac{c_i - \lambda}{w} \rfloor$$

In Algorithm 2, we assign the links into trees as follows. Let ℓ'_i be the number of unassigned links of node i and q_k be the number of links we need to assign into the k-th tree. At the beginning, we set $\ell'_i = \ell_i$ (line 1) and $q_k = n - 1$ (line 2). We repeat the following steps until all trees are assigned ($q_k = 0, \forall k \in [d]$) or there is no unassigned links left ($\ell'_i = 0, \forall i \in V$). First we choose the k-th with the the highest number of links need to be assigned (line 5) and the node i with the highest number of unassigned links ℓ'_i (line 6). Then, we assign $nl = \min(\ell'_i, q_k, \lfloor \frac{n}{4} \rfloor)$ of node i to the k-th tree. This way, we ensure that the number of assigned links of each node i does not exceed ℓ_i. Plus, different in the number of link need to be assigned in any trees does not exceed $\lfloor \frac{n}{4} \rfloor$. Hence, we can avoid the cases where we assign all of non-single-link nodes in some trees and have to fill up other trees with single-link nodes.

3.3.1.3 Broadcast Trees Construction

Intuitively, in each tree, the nodes that assigns more links is closer to the root nodes (which is the node that assigns the most links).

In detail, we build the k-th broadcast tree T_k in Algorithm 3 as follows. At line 2, we sort the nodes in V as $h_1, \cdots, h_n$ in the decreasing order of s_{k,h_i}, i.e.,

Algorithm 3: BuildBroadcastTrees procedure.

Input : Given the list of d trees S.

Output: Return d broadcast trees $T = (T_1, \cdots, T_d)$.

```
1 for k = 1 to d do
2     Sort h_1, h_2, ..., h_n such that s_{k,h_i} ≥ s_{k,h_{i+1}}, ∀i ∈ [n − 1]
3     Initiate an empty tree T_k with the root r_k = h_1
4     child = 1
5     for i = 1 to n do
6         for j = 1 to s_{k,h_i} do
7             Add (h_i, h_{child+j}) to T_k
8         child = child + s_{k,h_i}
9 Return T
```

$s_{k,h_i} \geq s_{k,h_{i+1}}, \forall i \in [n-1]$. At line 3, we set h_1 as the root node of T_k. For each node i, we add $s_{k,i}$ directed links from i to the first $s_{k,i}$ nodes that do not directed links from any node $j < i$. This way, we can guarantee that each node have at exactly one directed links from other node. Furthermore, since h_i is sorted in the decreasing order of s_{k,h_i}, we can ensure that all nodes can be reached from the root.

3.3.1.4 Transmission Schedule

After building the broadcast tree, the root nodes of those trees gather the data from the source node and then distribute to all other nodes through the broadcast trees.

In Algorithm 4, each node i assigns a *broadcast load* π_{k_i} to the tree T_k. Most of the time, each node i only assigns *one non-zero broadcast load* to a tree; all other tree will be assign zero broadcast load. Then, node i send the broadcast load to the

Algorithm 4: ConstructSchedule procedure

Input : Given the arrival rate λ, the list of broadcast tree T, and the parameter w.

Output: Return the transmission schedule f.

$\pi_{k,i} = 0, \forall k \in [d], i \in [n]$

$\eta_k = w, \forall k \in [d]$

$\lambda' = \{\lambda'_i = \lambda_i\}_{i \in [n]}$

while $\exists \eta_k > 0$ *and* $\exists \lambda'_i > 0$ **do**

    $k \leftarrow \arg\max_{k \in [d]} \eta_k$, $i \leftarrow \arg\max_{i \in [n]} \lambda'_i$

    $nl \leftarrow \min(\eta_k, \lambda'_i)$

    $\pi_{k,i} = nl$, $\eta_k = \eta_k - nl$, $\lambda'_i = \lambda'_i - nl$

for *each* $\pi_{k,i} > 0$ **do**

    $r_k =$ root of T_k, $T_k^i = T_k$

    if $r_k \neq i$ **then**

        Remove the (directed) edge to i from $T_k^{(i)}$

        Add (i, r_k) to $T_k^{(i)}$

    Set i as the root of $T_k^{(i)}$

    for *each node* $j \in V \setminus \{i\}$ **do**

        for *each link* (u, v) *on the path from* i *to* j *on* $T_k^{(i)}$ **do**

            $f_{uv}^{(i,j)} = f_{uv}^{(i,j)} + \pi_{k,i}$

Return f

roots of the trees. Then, the data will be forwarded through the broadcast tree.

Assign broadcast load. For each node i, we assign an amount $\pi_{k,i}$ of data to be broadcast through the tree T_k. Unless the root node of the broadcast tree already gather to much data (close to w), each node i only broadcast the data through one broadcast tree. From line 4-10, for each iteration in the while loop, we select node i with maximum (remaining) arrival rate and the tree k with maximum (remaining) capacity and set $\pi_{k,i}$ as the minimum of those values. This way, for we can guarantee that the each root node of the tree does not gather more than the capacity w amount of data.

Forwarding data through broadcast tree. For each source node i and tree T_k such that the broadcast load $\pi_{k,i} > 0$, nodes forward through the T_k an $\pi_{k,i}$ amount of data from node i. In detail, node i first forward $\pi_{k,i}$ amount of data to the root node r_k of T_k. Then the data will be send from r_k to all other node through T_k. Note that, since i already has the data, nodes will node upload the data to i again.

Let $T_k^{(i)}$ be the tree constructed from T_k by adding a new edge (i, r_k) removing the edge to node i. For each source node $j \neq i$, for all link (u, v) on the path from i to j on the tree $T_k^{(i)}$, we set the data flow from i to j on link (u, v) as $f_{uv}^{(i,j)} = \pi_{k,i}$.

3.3.2 ProSHeN$^+$: Distributed Data Distribution scheme

Inspired by ProSHeN, we develop ProSHeN$^+$ algorithm, a distributed data distribution scheme. This approach comprises of two key components. The first part involves the distributed construction of a network topology by nodes. To optimize the performance, we have adopted the same-capacity links principle to construct a topology such that the number of outgoing links is proportional to the node's uploading bandwidth. The second part entails the propagation of data across the network using a method similar to Bitcoin's propagation technique [37]. Specifically, nodes

utilize an invite-getdata-send method to avoid sending duplicated data. However, unlike Bitcoin, the communication links in ProSHeN$^+$ are half-duplex, i.e., nodes only forward data through the outgoing links.

3.3.2.1 Topology construction

Node constructs the network topology using a parameter w. Under this scheme, the capacity of each link is expected to be w. As a result, the number of outgoing connections a node creates is proportional to its upload capacity. For instance, if a node u has an upload capacity of C_u, it will create $d_u = \lfloor \frac{C_u}{w} \rfloor$ outgoing links. Node u will randomly select d_u outgoing neighbors and connect to them.

The ProSHeN$^+$ protocol is more robust in a heterogeneous network because the number of outgoing connections is proportional to the upload capacities of the nodes. This allows us to utilize nodes with high bandwidth to forward more data to other nodes in the network. In contrast, if all nodes have the same number of outgoing links (as in the design of Bitcoin), nodes with high bandwidth will be idle most of the time because they can forward data to all of their neighbors quickly.

Furthermore, when node u creates an outgoing connection to a node v, we refer to node u as an incoming neighbor of node v. This relationship can have important implications for the behavior of the network. Most of the time, nodes only forward the data to their outgoing neighbors.

3.3.2.2 Propagation method

ProSHeN$^+$ uses the invite-getdata-send propagation method, which is similar to Bitcoin and was described by Decker and Wattenhofer [37]. This method is efficient because it avoids sending duplicate data and enables nodes to selectively request the transactions or blocks they need. However, it may result in some redundancy in data

transmission by sending invite and getdata messages.

In contrast to Bitcoin, nodes in ProSHeN$^+$ use half-duplex links instead of full-duplex links. This means they only forward data through outgoing links, reducing the amount of redundant data, such as invite and getdata messages, that need to be transmitted. This approach increases the total amount of meaningful data, such as blocks and transactions, that are propagated in the network. As a result, the ProSHeN$^+$ protocol can handle a higher level of DoS attacks.

In summary, while ProSHeN$^+$ and Bitcoin share a similar propagation method, the former improves upon the latter by using half-duplex links to reduce redundancy and increase the amount of meaningful data propagated in the network. This enables the ProSHeN$^+$ protocol to handle a higher level of DoS attacks and provides a more efficient solution for blockchain networks.

When a node u in a peer-to-peer network receives a new message, it sends an invite message to all its outgoing neighbors. This message requests the receiver to send a specific piece of data to the sender. If node u is the source of the message, it sends the invite messages to all incoming and outgoing neighbors. This ensures that nodes in the network are notified of the new message quickly. When node v receives an invite message from node u that it doesn't have locally, it issues a getdata message to the sender of the invite message. This getdata message requests the actual data that the sender is asking for. Once received, node u will send the message to node v. This process is crucial for the proper functioning of the blockchain peer-to-peer network. It enables efficient and reliable sharing of information between nodes.

Moreover, the propagation scheme can also be useful when the network is under DoS attacks and the total arrival rate exceeds the network's capacity. In such cases, nodes can prioritize the propagation of important information, such as blocks or transactions included in compact blocks. This allows nodes to synchronize the data

on the blockchain and still achieve consensus.

3.4 Analysis

In this section, we perform an analysis for ProSHeN algorithm. We first prove that our broadcast trees overlay network is feasible (i.e., the total capacity in all upload-links from a node do not exceed its upload capacity). Then, we show that the broadcast trees overlay network can achieve $1 - \frac{1}{d} - \frac{2}{n}$ optimal throughput and $O(\log n)$ latency, while satisfying the sparsity constraint.

3.4.1 Near-optimal throughput

Lemma 35. *Consider a set of n nodes V, upload capacity C, a vector throughput λ, and parameter d, w such that*

$$d \leq \frac{1}{n-1}\left(\sum_{i \in V} \ell_i\right), \tag{3.11}$$

where $\ell_i = \lfloor \frac{c_i - \lambda_i}{w} \rfloor$. Then AssignTrees *algorithm returns a list S of d tree that satisfies the following conditions.*

$$\sum_{i \in V} s_{k,i} = n-1, \forall k \in [d]. \tag{Eq.3.9}$$

$$\sum_{k \in [d]} s_{k,i} \leq \ell_i, \forall i \in V. \tag{3.12}$$

Proof. For simplicity, without other mentioned, we only refer to the lines Algorithm 2 in this proof. First, we show that S satisfies Eq.3.9. We consider the termination conditions of the while loop from line 4 as follows.

- $q_k = 0, \forall k \in [d]$. Before the while loop starts, at line 2, we set $q_k = n - 1$. Plus, from line 8-9, we only subtract q_k by nl after we add nl to some $s_{k,i}$. Thus, we

have,

$$\sum_{i \in V} s_{k,i} = n - 1.$$

- $\ell'_i = 0, \forall i \in V$ and $\exists q_k > 0$. We show that this case cannot happen if Eq. 3.11 is satisfied. From line 9-10, we only subtract ℓ'_i by nl after we subtract q_k by nl. Plus, before the while loop starts, we set $q_k = n-1, \forall k \in [d]$ and $\ell'_i = \ell_i, \forall i \in V$. Thus, before the while loop starts, we have,

$$\sum_{i \in V} \ell_i < \sum_{k \in [d]} q_k = d(n-1).$$

  Recall from line 1, $\ell_i = \lfloor \frac{c'_i}{w} \rfloor = \lfloor \frac{c_i - \lambda_i}{w} \rfloor$. Combine with Eq. 3.11, we have, $\sum_{i \in V} \ell_i \geq d(n-1)$ (contradiction).

Now, we show that S satisfies Eq.3.12. From line 8-10, we only add nl to $s_{k,i}$ when we subtract ℓ_i by nl. Thus, the sum of all $s_{k,i}$ when the while loop ends cannot exceed ℓ_i before the while loop starts. Hence, S satisfies Eq.3.12.

□

Lemma 36. *Consider a set of n nodes V and a list of d trees S that satisfies Eq.3.9. Then* BuildBroadcastTrees *algorithm returns a list T of d trees in which for each tree T_k, all nodes in V is reachable from the root node r_k of the tree T_k.*

Proof. For simplicity, without other mentioned, we only refer to the lines Algorithm 3 in this proof.

Consider the k-th iteration of the first for loop to build the tree T_k. Let $p_i = 1 + \sum_{j \in [i]} s_{k,h_j}$. Note that, since $\sum_{i \in V} s_{k,h_i} \geq n - 1$ (Eq.3.9) and $s_{k,h_i} \geq s_{k,h_{i+1}}$ (line 2 in the algorithm), $\forall i \in [n-1]$, we have,

$$i + 1 \leq p_i \leq n, \forall i \in [n-1] \text{ and } p_n = n.$$

We will show by induction that after the i-th iteration of the second for loop, the list of nodes $h_1, \cdots, h_{p_i}$ are reachable from the root node $r_k = h_1$.

- After the first iteration, the root node h_1 makes s_{k,h_i} connections to $h_2, \cdots, h_{p_1}$ (line 6-7). Thus, the list of nodes $h_1, \cdots, h_{p_1}$ are reachable from h_1.
- Now, we assume that $h_1, \cdots, h_{p_i}$ are reachable from h_1 after the i-th iteration. We will show that $h_1, \cdots, h_{p_{i+1}}$ are reachable from h_1 after the $(i+1)$-th iteration. Indeed, in the $(i+1)$-th iteration h_{i+1} make connections to $s_{k,h_{i+1}}$ nodes $p_i + 1, \cdots, p_{i+1}$. Node that, after i-th iteration, h_{i+1} is reachable from h_1 (since $p_i \geq i+1$). Hence, after $(i+1)$-th iteration, $h_{p_i+1}, \cdots, h_{p_{i+1}}$ are reachable from h_1.

□

Lemma 37. *Consider a set of n nodes V, upload capacity C, a vector throughput λ, and parameter d, w, that satisfy Eq.3.9, Eq.3.12, and*

$$\sum_{i \in V} \lambda_i \leq w \times d. \tag{3.13}$$

Then BuildSchedule *algorithm produces a transmission schedule f with arrival rate λ that satisfies the flow's condition in Eq.3.1.*

Proof. For simplicity, without other mentioned, we only refer to the lines Algorithm 4 in this proof.

Consider a soure node i and a broadcast tree T_k. We denote the data on an edge (u, v) of the flow from a source node i to a target node j in the broadcast tree T_k as $f_{u,v}^{(i,j)(k)}$. From lines 11-14, $f_{u,v}^{(i,j)(k)}$ can be either $\pi_{k,i}$ or 0.

The data on an edge $(u, v) \in E$ of the flow from i to j is the total data on all

the broadcast trees, i.e.,

$$f_{u,v}^{(i,j)} = \sum_{k \in [d]} f_{u,v}^{(i,j)(k)}$$

To prove the flow's constraint, we show that on each tree T_k, the data flow from the source node i to all target node j is $\pi_{k,i}$, i.e.,

$$\begin{aligned} F_u^{(i,j)(k)} = &\sum_{v \in V} f_{vu}^{(i,j)(k)} - \sum_{v \in V} f_{uv}^{(i,j)(k)} + \pi_{k,i} \mathbb{1}_{\{u=i\}} \\ &- \pi_{k,i} \mathbb{1}_{\{u=j\}} = 0, \forall u \in V, \forall i \in V, j \in V \setminus \{j\}, k \in [d]. \end{aligned}$$

Indeed, for any source node $i \in V$ and any target node $j \in V \setminus \{i\}$, based on lines 18-19, we have

$$f^{(i,j)(k)} u, v = \begin{cases} \pi_{k,i}, \text{ if } (u,v) \text{ on the path from } i \text{ to } j \text{ on } T_k^{(i)}, \\ 0, \text{ otherwise.} \end{cases}$$

Note that, in the tree $T_k^{(i)}$, there is exactly on path from any source node i to any target node j. Thus, $F_u^{(i,j)(k)} = 0$, $\forall u \in V$.

Thus, we can conclude that data flow from the source node i to all target node j on the tree T_k is $\pi_{k,i}$. We show that $\sum_{k \in [d]} \pi_{k,i} = \lambda_i$ by contradiction. We consider the two following cases.

- $\sum_{k \in [d]} \pi_{k,i} > \lambda_i$. From line 8-10, we only add nl to $\pi_{k,i}$ if we subtract nl from λ_i'. When the while loop from lines 4-10 terminated, we have $\lambda_i' \geq 0$. Plus, before the while loop starts, we set $\lambda_i' = \lambda_i$. Thus, when the while loop terminated, we have, $\lambda_i = \sum_{k \in [d]} \pi_{k,i} + \lambda_i' \geq \sum_{k \in [d]} \pi_{k,i}$ (contradiction).
- $\sum_{k \in [d]} \pi_{k,i} < \lambda_i$. In this case, when the while loop terminated, $\lambda_i' > 0$ (since $\lambda_i = \sum_{k \in [d]} \pi_{k,i} + \lambda_i'$). Thus, the while loop terminated when $\eta_k = 0, \forall k \in [d]$.

From line 9-10, subtract a η_k by nl before we subtract a λ'_i by nl. Thus, before the while loop starts, we have $\sum_{i\in V}\lambda_i > \sum_{k\in[d]}\eta_k = d\times w$. This contradict with Eq.3.13.

Finally, we show that f satisfies the flow's constraint in Eq.3.1 for any node u as follow.

$$\begin{aligned}
&\sum_{v\in V} f_{vu}^{(i,j)} - \sum_{v\in V} f_{uv}^{(i,j)} + \lambda_i \mathbb{1}_{\{u=i\}} - \lambda_i \mathbb{1}_{\{u=j\}} \\
=&\sum_{v\in V}\sum_{k\in[d]} f_{vu}^{(i,j)(k)} - \sum_{v\in V}\sum_{k\in[d]} f_{uv}^{(i,j)(k)} + \left(\sum_{k\in[d]}\pi_{k,i}\right)\mathbb{1}_{\{u=i\}} \\
&\qquad\qquad - \left(\sum_{k\in[d]}\pi_{k,i}\right)\mathbb{1}_{\{u=j\}} \\
=&\sum_{k\in[d]}\left(\sum_{v\in V} f_{vu}^{(i,j)(k)} - \sum_{v\in V} f_{uv}^{(i,j)(k)} + \pi_{k,i}\mathbb{1}_{\{u=i\}} - \pi_{k,i}\mathbb{1}_{\{u=j\}}\right) = 0.
\end{aligned}$$

□

Lemma 38. *Consider a set of n nodes V, upload capacity C, a vector throughput λ, and parameter d, w, that satisfy Eq.3.9, Eq.3.12, and*

$$\sum_{i\in V}\lambda_i \le w\times d. \tag{3.14}$$

Then BuildSchedule *procedure (Algorithm 4) produces a transmission schedule f with arrival rate λ that satisfies the capacity's condition in Eq.3.4.*

Proof. From Eq.3.2, we have,

$$\begin{aligned}
f_{uv}^{(i)} &= \max_{j\in V} f_{uv}^{(i,j)} = \max_{j\in V}\left(\sum_{k\in[d]} f_{uv}^{(i,j)(k)}\right) \\
&\le \sum_{k\in[d]}\left(\max_{j\in V} f_{uv}^{(i,j)(k)}\right).
\end{aligned}$$

From lines 13-18, if $(u,v) \notin T_k^{(i)} \subseteq T_k \cup \{(i,r_k)\}$ (r_k is the root of T_k), $f_{uv}^{(i,j)(k)} = 0$. Otherwise, $f_{uv}^{(i,j)(k)}$ can be at most $\pi_{k,i}$. Thus, we have,

$$f_{uv}^{(i)} \leq \sum_{k\in[d]:(u,v)\in T_k\cup\{(i,r_k)\}} \pi_{k,i}.$$

Replace this the right-hand side of Eq.3.3, we have

$$\begin{aligned} g_{uv} = \sum_{i\in V} f_{uv}^{(i)} &\leq \sum_{i\in V} \left(\sum_{k\in[d]:(u,v)\in T_k\cup\{(i,r_k)\}} \pi_{k,i} \right) \\ &= \sum_{k\in[d]:(u,v)\in T_k} \left(\sum_{i\in V} \pi_{k,i} \right) + \sum_{k\in[d]} \left(\mathbb{1}_{v=r_k} \pi_{k,u} \right). \end{aligned}$$

Note that, from line 4-10 of algorithm 4, we have

$$\sum_{k\in[d]} \pi_{k,i} = \lambda_i, \forall i \in V,$$

$$\sum_{i\in V} \pi_{k,i} \leq w, \forall k \in [d],$$

We now show that f satisfies the capacity's constraint for in Eq.3.4 any node u as follows.

$$\begin{aligned} \sum_{v\in V} g_{uv} &\leq \sum_{v\in V} \left(\sum_{k\in[d]:(u,v)\in T_k} \left(\sum_{i\in V} \pi_{k,i} \right) + \left(\mathbb{1}_{v=r_k} \pi_{k,u} \right) \right) \\ &\leq \sum_{v\in V} \left(\sum_{k\in[d]:(u,v)\in T_k} w \right) + \sum_{k\in[d]} \left(\pi_{k,u} \right) \\ &= \sum_{k\in[d]} \sum_{v\in V:(u,v)\in T_k} w + \lambda_u = \sum_{k\in[d]} s_{k,u} \times w + \lambda_u \\ &\leq \ell_u \times w + \lambda_u \leq c'_u + \lambda_u = c_u. \end{aligned}$$

□

Lemma 39. *For any $d \geq 1$ and any arrival rate λ such that*

$$\sum_{i \in V} \lambda_i \leq \frac{1}{(1 + \frac{1}{n-1} + \frac{2}{d})} \mathsf{OPT}_{TP}, \tag{3.15}$$

where $\mathsf{OPT}_{TP} = \frac{\sum_{i \in V} c_i}{n-1}$ is optimal throughput. Set w as the maximum value that satisfies

$$\sum_{i \in V} \lfloor \frac{c_i - \lambda_i}{w} \rfloor \geq d(n-1). \tag{3.16}$$

ProSHeN algorithm returns a feasible transmission schedule f of λ.

Proof. Combine with Lemma 37 and Lemma 38, ProSHeN algorithm returns a feasible transmission schedule f of λ if

$$\sum_{i \in V} \lambda_i \leq dw.$$

We now show that the above equation is satisfied if w is is the maximum value that satisfies Eq.3.16. We have,

$$\sum_{i \in V} \lfloor \frac{c_i - \lambda_i}{w} \rfloor < (d+1)(n-1)$$

Furthermore, there exists $i \in V$, such that $\lfloor \frac{c_i - \lambda_i}{w} \rfloor = \frac{c_i - \lambda_i}{w}$. Indeed, let $x = \min_{i \in V} \left(\frac{c_i - \lambda_i}{w} - \lfloor \frac{c_i - \lambda_i}{w} \rfloor \right)$. We prove $x = 0$ by contradiction. Assume $x > 0$, let $j = arg\min_{i \in V} \left(\frac{c_i - \lambda_i}{w} - \lfloor \frac{c_i - \lambda_i}{w} \rfloor \right)$. Let $w' = \frac{c_j - \lambda_j}{\lfloor \frac{c_j - \lambda_j}{w} \rfloor}$. We have, $\lfloor \frac{c_i - \lambda_i}{w} \rfloor = \lfloor \frac{c_i - \lambda_i}{w'} \rfloor$, $\forall i \in V$. Plus, since $\frac{c_j - \lambda_j}{w} > \lfloor \frac{c_j - \lambda_j}{w} \rfloor$, $w' > w$. Hence, there exists $w' > w$ that satisfies Eq.3.16. Therefore, we have,

$$\begin{aligned} \sum_{i \in V} \frac{c_i - \lambda_i}{w} &< \sum_{i \in V \setminus j} \left(\lfloor \frac{c_i - \lambda_i}{w} \rfloor + 1 \right) + \lfloor \frac{c_j - \lambda_j}{w} \rfloor \\ &= \sum_{i \in V} \lfloor \frac{c_i - \lambda_i}{w} \rfloor + (n-1) < (d+2)(n-1). \end{aligned}$$

Multiple both sides by $\frac{w}{n-1}$, we have,

$$\frac{1}{n-1}\left(\sum_{i\in V}(c_i-\lambda_i)\right)\leq(d+2)w$$
$$\Rightarrow \mathsf{OPT}_{TP}-\frac{1}{n-1}\left(\sum_{i\in V}\lambda_i\right)\leq(d+2)w$$

Using Eq. 3.15, we have

$$\Rightarrow\left(1+\frac{1}{n-1}+\frac{2}{d}-\frac{1}{n-1}\right)\left(\sum_{i\in V}\lambda_i\right)\leq(d+2)w$$
$$\Rightarrow\sum_{i\in V}\lambda_i\leq dw$$

Combine with Lemma 37 and Lemma 38, ProSHeN algorithm returns a feasible transmission schedule f of λ if $\sum_{i\in V}\lambda_i\leq dw$.

□

3.4.2 $O(\log n)$ latency

Now, we prove that, all the broadcast trees have $O(\log n)$ depth (i.e., the longest distance from the root to other nodes is $O(\log n)$).

Lemma 40. *Consider a set of $n\geq 8$ nodes V, upload capacity C, a vector throughput λ, and parameter d, w such that*

$$\sum_{i\in V}\lfloor\ell_i\rfloor\geq d(n-1)\ \textit{and}\ d\geq 2d\leq\frac{1}{n-1}\left(\sum_{i\in V}\ell_i\right)$$

Let S be the list of d trees that is returned by AssignTrees *procedure (Algorithm 2). For the k-th tree, we say $i\in V$ is a single-link node if $s_{k,i}=1$. Then, for any tree*

S_k, the number of single-link nodes is at most $\frac{3n}{4}$, i.e.,

$$\sum_{i \in V : s_{j,i}=1} s_{j,i} \leq \frac{3n}{4} \tag{3.17}$$

Proof. Assume toward contradiction that, $\sum_{i \in V : s_{j,i}=1} s_{j,i} > \frac{3n}{4}$. We define an *adding single-link node event* is the event where a single upload link is added to a tree. Consider the first adding single-link node event occurs when node i add the new upload link to tree $S_{j'}$. We consider two cases as follows.

- The number upload-links need to be added to $S_{j'}$ is 1. From algorithm 2, $S_{j'}$ is the tree that is needed to add the most upload-links. Thus, all tree has at most 1 upload-link need to be added.

  Since, this is the first adding single-link node event, all nodes either add more than 1 links to the trees or add nothing. Hence, we have, $\sum_{i \in V : s_{j,i}=1} s_{j,i} \leq 1 < \frac{3n}{4}, \forall j \in [d]$. This contradict the our assumption.

- Node i only have 1 remaining upload-links. From algorithm 2, i is the node have the most upload-links. Thus, all nodes have at most 1 upload links. In other words, the total remaining upload-links of all nodes is at most n.

  Recall that, $S_{j'}$ is the tree that is needed to add the most upload-links. Since we assume there exists a tree S_j that has more than $\frac{3n}{4}$ nodes that have 1 upload-link, $S_{j'}$ also has more than $\frac{3n}{4}$ nodes that have 1 upload-link. Further, each time, a node can only add at most $\frac{n}{4}$. Therefore, all other tree have more than $\frac{3n}{4} - \frac{n}{4} = \frac{n}{2}$ nodes that have 1 upload-link. Hence, the number of upload-links need to be added since the first adding single-link node event is more than $\frac{n}{2} \times d \geq n$ (contradiction).

□

Lemma 41. *Given a tree S_k that satisfies $\sum_{i\in V:s_{k,i}>1} s_{k,i} \geq \frac{n}{4}$ (equivalent with Eq. 3.17) and $\sum_{i\in V} s_{k,i} = n-1, \forall k \in [d]$ (Eq.3.9). Algorithm 3 returns list T of d broadcast trees with $O(\log n)$ depth.*

Proof. Consider the k-th iteration of the first for loop to build the tree T_k. Let $T_{k,i}$ be the subtree of T_k that consists of i nodes $h_1, \cdots, h_i$ ($T_{k,1} \subset T_{k,2} \subset \cdots \subset T_{k,n} = T_k$). We denote $\mathsf{depth}(T_{k,i})$ as the depth (the maximum distance from the root node h_1 to all other nodes) of the tree $T_{k,i}$. $\mathsf{depth}(T_{k,1}) = 0$.

Let $p_i = 1 + \sum_{j\in[i]} s_{k,h_j}$. We have $p_1 \leq p_2 \leq \cdots \leq p_n = n$.

From lines 6-7, node h_i, make s_{k,h_i} connections to $h_{p_{i-1}+1}, \cdots, h_{p_i}$. Thus,

$$\mathsf{depth}(T_{k,p_i}) \leq \mathsf{depth}(T_{k,i}) + 1$$

Recall from line 2 that $s_{k,h_i} \geq s_{k,h_{i+1}}$, $\forall i \in [n-1]$. Let $\tau \in [n]$ be the biggest index such that $s_{k,h_\tau} > 1$, and τ_1 be the biggest index such that $s_{k,h_{\tau_1}} \geq 1$. We have

$$p_\tau \geq \frac{n}{4} (\text{Eq. 3.17}) \text{ and } p_{\tau_1} = n \text{ (Eq.3.9)}.$$

For any $i \in [\tau]$, we have,

$$p_i = 1 + \sum_{j\in[i]} s_{k,h_j} \geq 1 + 2i$$

Thus, the depth of T_{k,p_τ} is $\mathsf{depth}(T_{k,p_\tau}) = O(\log \tau) = O(\log n)$.

For $i \leq \tau_1$, we have,

$$p_i - i = p_{i-1} - (i-1) + (s_{k_i} - 1) \geq p_{i-1} - (i-1).$$

Thus, for any i such that, $\tau \leq i \leq \tau_1$, we have,

$$p_i - i \geq p_\tau - \tau \geq \frac{p_\tau}{2} \geq \frac{n}{8}$$

(Here, $\tau \leq \frac{p_\tau}{2}$, thus, $p_\tau - \tau \geq \frac{p_\tau}{2}$.)

$$\Rightarrow \mathsf{depth}(T_{k,i+\lfloor \frac{n}{8} \rfloor}) \leq \mathsf{depth}(T_{k,p_i}) \leq \mathsf{depth}(T_{k,i}) + 1$$

Thus, for any i, j such that $\tau \leq i \leq j \leq p_{\tau_1} = n$, we have

$$\mathsf{depth}(T_{k,j}) \leq \mathsf{depth}(T_{k,i}) + \frac{j-i}{\lfloor \frac{n}{8} \rfloor} \leq \frac{n}{\lfloor \frac{n}{8} \rfloor} = \mathsf{depth}(T_{k,i}) + O(1).$$

Let $i = p_\tau$ and $j = n$, we have,

$$\mathsf{depth}(T_{k,n}) = \mathsf{depth}(T_{k,p_\tau}) + O(1) = O(\log n) + O(1).$$

□

Lemma 42. *For any source $i \in V$ and any broadcast tree $T_k \in T$, the depth of the tree $T_k^{(i)}$ is $O(\log n)$*

3.4.3 Sparsity constraint

Lemma 43. *Consider a set of n nodes V, and parameter d. ProSHeN returns a transmission schedule f with at most $(d+1)n$ non-zero links.*

Proof. Let $E_f = \{(u, v) : \exists i, j \in V \text{ such that } f_{uv}^{(i,j)} > 0\}$. There are two types of links in E_f as follows.

- $\exists k \in [d]$ such that $(u, v) \in T_k$. From Algorithm 3, each tree T_k has $(n-1)$ links. Thus, there are at most $d(n-1)$ pairs (u, v) that satisfy this conditions.
- $\exists k \in [d]$ such that $\pi_{k,u} > 0$ and v is the root of T_k. Form the while loop in lines 4-10 in Algorithm 4, the number of pair k, u such that $\pi_{k,u} > 0$ equals the

number of iterations of the while loops. From lines 5-6, for each iteration, we first select the values of $\eta_k > 0$ ($k \in [d]$) and $\lambda'_i > 0$ ($i \in [n]$). Then, either (or both) η_k or λ'_i will be reduce to 0 (lines 7-10). Since the while loop terminated when all $\eta_k = 0$ ($k \in [d]$) or all $\lambda'_i = 0$ ($i \in [n]$), the number of iterations is at most $n+d$. Thus, there are at most $n+d$ pairs (u, v) that satisfy this conditions.

Therefore, the maximum number of links in E is at most $(n-1)d + n + d = n(d+1)$. □

3.5 Experiments

We provide the performance evaluation of ProSHeN$^+$ and the data distribution scheme of Bitcoin. We also consider ProSHeN* that use the multi-tree topology of ProSHeN (see Algorithm 3) and the transmission scheduling of ProSHeN$^+$.

3.5.1 The setup for experiments

Blockchain simulator. We build an *event management data structure* that allows us to simulate the mining and data propagation among nodes, by processing/updating events.

- *Simulated mining.* We simulate the mining using a Bernoulli random variable for each node in each time unit (0.1s). In each time unit, each node has a probability p of generating a new block. The probability p so that a new block is generated in every 10 minutes.

- *Transaction generation.* For every time unit, some transactions are randomly generated among the nodes based on the transaction rate.

- *Compact block relay [29].* Compact Block Relay is a feature implemented in the Bitcoin that allows for more efficient and faster transmission of blocks between

nodes in the Bitcoin network. Instead of sending the full block, node constructs and sends a compact block by replacing each transaction with a 6-byte non-cryptographic hash of that transaction's identifier.

Topology construction. We construct the topology for a $5,000$ nodes networks in which the average outgoing links of each node is 8 (the default number of outgoing links in Bitcoin [37]).

- Bitcoin. We construct the topology of Bitcoin network as described in [37]. Each node in the network randomly selects 8 other nodes to request establishing outgoing transactions.
- ProSHeN^{+}. We construct the topology as described in Section 3.3.2. Each node requests establishing outgoing to other nodes based on its bandwidth.
- ProSHeN*. We construct the multi-tree topology as described in Algorithm 3.

Propagation method. Nodes propagate data using invite-getdata-send method to avoid sending duplicated data.

- Bitcoin. A node sends invite messages through both incoming and outgoing links.
- ProSHeN^{+} and ProSHeN*. If a node is the source of the data, it sends invite messages through both incoming and outgoing links. Otherwise, the node only sends the invite message through the outgoing links.

Bandwidth distribution. Based on the provisioned bandwidth statistics in [49], we fit the bandwidth distribution using a log-normal distribution [57]. Here, the average bandwidth of a node is 73.1 Mb/s. We have the Gini coefficient [43] to measure the level of heterogeneity for the bandwidth distribution. The Gini coefficient

ranges from 0 to 1, with 0 representing perfect equality (where every node has the same bandwidth) and 1 representing perfect inequality (where one node has all the bandwidth, while the remaining has nothing).

Parameter settings. We set the transaction size is 500 bytes[3] and block size is $4MB$ (the maximum block size of Bitcoin[4]).

3.5.2 Experiment results

Effect of heterogeneity. We first study the effect of heterogeneity on the three data distribution scheme. Here, we set the transaction rate to 1 Mb/s.

As shown in Figure 14a, when the heterogeneity (Gini coefficient) increases, the block propagation time of Bitcoin increases significantly. As a result, the relative time that nodes waste on not mining the longest chain also increases when the Gini coefficient increases (see Figure 14b). Specifically, when the Gini coefficient of the bandwidth distribution increases from 0 to 0.9, the block propagation time of Bitcoin increases from 8.8s to 43.8s, the relative time that nodes waste on not mining the longest chain increase from 0.87% to 7.23%.

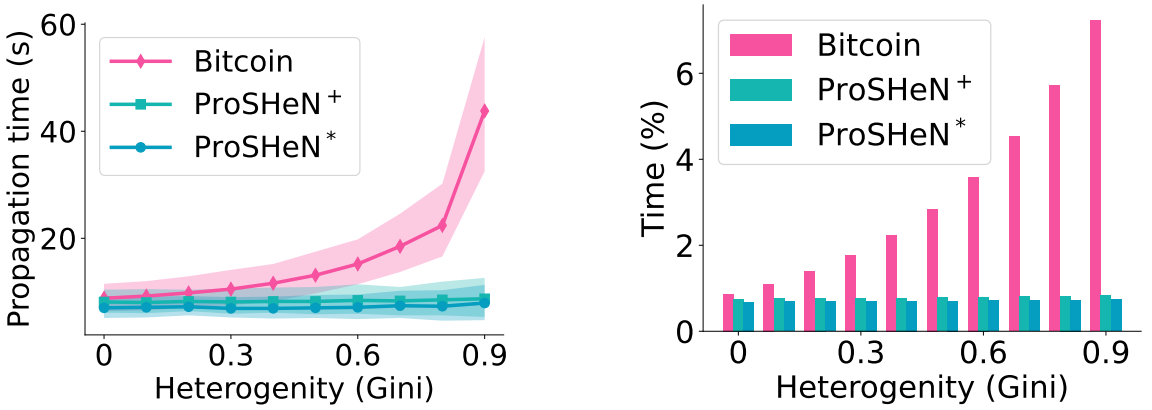
On the other hand, the block block propagation time and the relative time that nodes waste on not mining the longest chain of ProSHeN$^+$ and ProSHeN* remain stable when the Gini coefficient increases.

DoS (transaction flooding) attacks. We also study the effect of DoS (transaction flooding) attacks. We increase the transaction rate to 71.3 Mb/s (same as the average bandwidth).

As shown in Figure 15, the Bitcoin data distribution scheme suffers more under

[3] `https://bitcoinvisuals.com/chain-tx-size`

[4] `https://bitcoinmagazine.com/guides/what-is-the-bitcoin-block-size-limit`



(a) The block propagation time

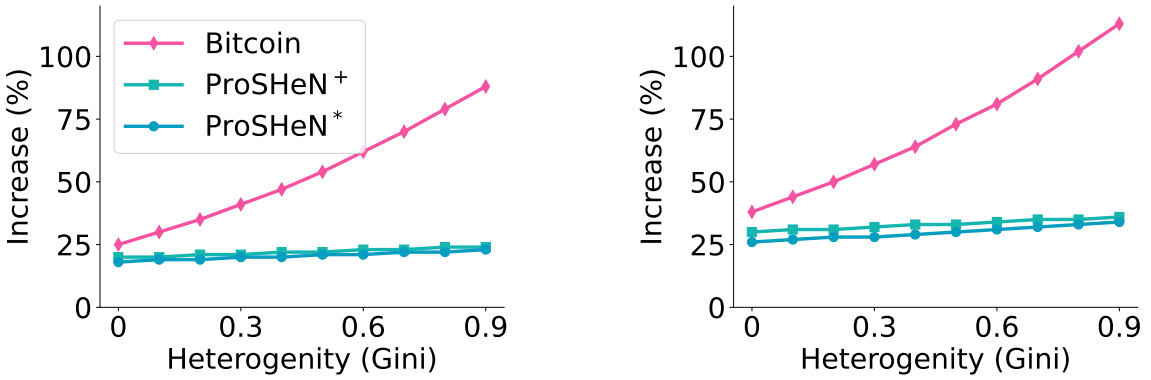
(b) The relative time that nodes waste on not mining the longest chain

Figure 14: The block propagation time and the relative time that nodes does not mine on the longest chain.

the DoS attack. For example, if the Gini coefficient of the bandwidth distribution is 0.9, under DoS attack, the block propagation time and the relative time that nodes waste on not mining the longest chain of Bitcoin increase by 88% and 113%, respective. Meanwhile, the number for ProSHeN$^+$ and ProSHeN* are 24%, 36%, and 23%, 34%, respective.

Double-spending attacks. We consider the HashSplit attacks [93] to perform double-spending attack. We consider an adversary that can instantly send and receive any data from any other node. The adversary also perform the DoS attack to reduce the requirement on the mining power.

As shown in Fig 16, when the Gini coefficient of the bandwidth distribution increases, the required mining power to perform double-spending attack on Bitcoin reduces significantly. When the Gini coefficient increase from 0.1 to 0.9, the required mining power for the adversary to revert a 6 confirmations transaction with a probability at least 1% reduce from 27.22% to 13.40%. Meanwhile, the required mining



(a) The block propagation time

(b) The relative time that nodes waste on not mining the longest chain

Figure 15: The increase under DoS attack of the block propagation time and the relative time that nodes waste on not mining the longest chain.

power to perform double-spending attack on ProSHeN$^+$ and ProSHeN* remains stable at approximately 28%.

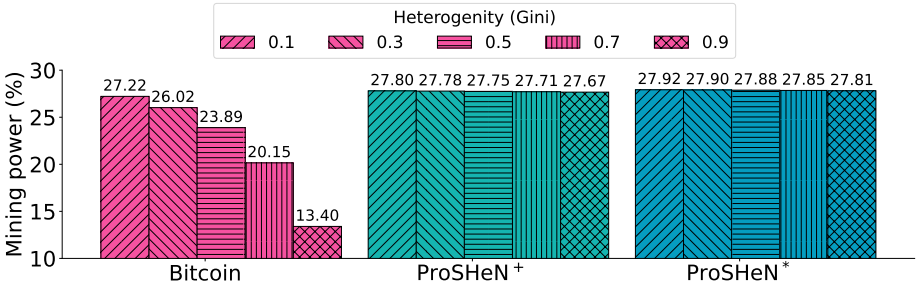


Figure 16: The required mining power by an adversary to reverse a transaction with 6 confirmations with a probability at least 1%.

CHAPTER 4

THE CONSENSUS LAYER

In this chapter, we first present an impossibility result for *all* proof-of-stake protocols under the *single-extension* design framework. In this framework, each honest player is allowed to extend exactly one chain in each round; the state-of-the-art permissionless PoS protocols (e.g., Praos, Snow White, and more), are all under this single-extension framework. Our impossibility result states that, if a single-extension PoS protocol achieves the best possible unpredictability, then this protocol cannot be proven secure unless more than 73% of stake is honest. Then, to overcome the impossibility result, we introduce a new design framework, called *multi-extension* PoS, which allows each honest player to extend *multiple* chains in a round. We develop a novel strategy called "D-distance-greedy" strategy (where D is a positive integer), in which honest players are allowed to extend *a set of best chains that are "close" to the longest chain.* (Of course, malicious players are allowed to behave arbitrarily in the protocol execution.) This "D-distance-greedy" strategy enables us to construct a class of PoS protocols that achieve the best possible unpredictability. Plus, we design a new tiebreak rule for the multi-extension protocol to chose the best chain that can be extended faster. This ensures that the adversary cannot slowdown the chain growth of honest players. Note, these protocols can be proven secure, assuming a much smaller fraction (e.g., 57%) of stake to be honest.

The chapter is organized as follows. In Section 4.1, we introduce an analytical framework for PoS protocols. In Section 4.2, we show an impossibility result for the *single-extension* PoS protocols. In Section 4.3, we construct a new PoS protocol in the

multi-extension design framework, bypassing the impossibility for the single-extension PoS protocols. Next, we provide the security analysis of our new PoS protocol. In Section 4.4, we provide an overview of the security analysis. In Section 4.5, we provide a new analysis framework to analyze the chain growth property of a PoS protocol in the multi-extension design framework. Then, in Section 4.6, we apply the new analysis framework to analyze the chain growth property for our new PoS protocol. In Section 4.7, we propose a new analysis framework to analyze the common prefix property for the new protocol. In Section 4.8, we show the chain quality and the best possible unpredictability properties of the new protocol. Finally, in Section 4.9, we discuss how to upgrade our protocol to a full-fledged blockchain protocol and show how to enable players to register their key-pairs adaptively. In Section 4.10, we provide the related work. Supplemental materials are presented in Section 4.11.

4.1 Security Model

4.1.1 Blockchain protocol executions

The security of Bitcoin-like PoW-based protocols has been rigorously investigated by Garay et al. [47] and then by Pass et al. [88] in the cryptographic setting. Below we define a framework for analyzing Bitcoin-like PoS-based blockchain protocols. We note many formulation ideas are taken from the previous frameworks [47, 88].

The execution of a PoS blockchain protocol. Following Canetti's formulation of the "real world" executions [21], we present an abstract model for a PoS blockchain protocol Π in the hybrid world of the partially synchronous network communication functionality, the random oracles, and certain initialization functionality.

We consider the execution of blockchain protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$, where κ is a security parameter. A necessary condition in all common

blockchain systems is that all players agree on the first, i.e., the *genesis block*. The genesis block consists of the identities (e.g., public keys) and the stake distribution of the players. Here, the registered players must control a certain number of stake. During the protocol execution, the stake distribution can be changed, the player can register to join or deregister to leave the system. For simplicity, we focus on the idealized "flat" model where all PoS-players have the same number of stake. In the non-flat model, the players that have more stake can register multiple identities.

The environment $\mathcal{Z}$ can "manage" protocol players through an adversary $\mathcal{A}$ that can dynamically corrupt honest players. More concretely, the protocol execution proceeds as follows. Each player in the execution is initialized with an initial state including all initial public information, e.g., a genesis block. The environment $\mathcal{Z}$ first activates the adversary $\mathcal{A}$ and a set $\mathcal{P}$ of PoS-players. The environment $\mathcal{Z}$ also provides instructions for the adversary $\mathcal{A}$. The execution proceeds in rounds, and in each round, a protocol player could be activated by the environment or the functionalities. Players are equipped with (roughly synchronized) clocks that indicate the current round.

In any round r, each PoS-player $P \in \mathcal{P}$, with a local state $state_r$, receives a message from $\mathcal{Z}$, and potentially receives messages from other players. Then, it executes the protocol, broadcasts a message to other players, and updates its local state. Note that, the network is controlled by the adversary, i.e., the adversary $\mathcal{A}$ is responsible for delivering all messages sent by players. The adversary $\mathcal{A}$ can reorder or delay the messages. However, it cannot modify the messages. Plus, any message, that is broadcasted by an honest player, is guaranteed to arrive at all other honest players within a maximum delay of Δ rounds.

At any round r of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, P)$, where $P \in \mathcal{P}$, to adversary $\mathcal{A}$. Then, the adversary $\mathcal{A}$ will have access to the player's local

state and control P. Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint $\texttt{VIEW}$ of all players (i.e., all their inputs, random coins and messages received, including those from the random oracle) in the above protocol execution; note that this joint view fully determines the execution.

Protocol players are allowed to join the protocol execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. In the current version of our modeling, we assume that when (honest) PoS-players leave the protocol execution, they will *erase* their own local internal information.[1]

Random oracles. As mentioned, we assume the availability of random oracles that capture the idealization of hash functions. Two hash functions are used for computing the context of the chains, and for creating the puzzles, respectively; note, to create a puzzle, the context of (certain) chains will be computed first.

Block and blockchain basics. A *blockchain* $\mathcal{C}$ consists of a sequence of ℓ concatenated blocks $B_0\|B_1\|B_2\|\cdots\|B_\ell$, where $\ell \geq 0$ and B_0 is the initial block (genesis block). We use $\mathsf{len}(\mathcal{C})$ to denote *blockchain length*, i.e., the number of blocks in blockchain $\mathcal{C}$; and here $\mathsf{len}(\mathcal{C}) := \ell$. (Note that since all chains must consist of the genesis block, we do not count it as part of the chain's length. In other words, a chain $\mathcal{C}$ with length ℓ actually has $\ell + 1$ blocks in total.) We use sub blockchain (or subchain) for referring to a segment of a chain; here for example, $\mathcal{C}[0, \ell]$ refers to an entire blockchain, whereas $\mathcal{C}[j, m]$, with $j \geq 0$ and $m \leq \ell$ would refer to a sub blockchain $B_j\|\cdots\|B_m$. We use $\mathcal{C}[i]$ to denote the i-th block, B_i in blockchain $\mathcal{C}$; here i denotes the *block height* of B_i in chain $\mathcal{C}$.

If blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}_1$, we write $\mathcal{C} \preceq \mathcal{C}_1$. If a chain $\mathcal{C}$ is truncated the last κ blocks, we write $\mathcal{C}[\neg\kappa]$. For some $\mathcal{A}, \mathcal{Z}$, consider some $\texttt{VIEW}$ in the support of $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. We use the notation $\texttt{VIEW}^r$ to denote the prefix of $\texttt{VIEW}$

[1]Players may sell their own secret keys; this is out of scope of this paper.

up until round r. Let $\mathbb{C}^r$ be the set of chains in $\texttt{VIEW}^r$, and let $\mathcal{C}_i^r$ be the chain in the view of player i at round r.

4.1.2 Chain growth, common prefix, and chain quality

Previously, several fundamental security properties for Bitcoin-like PoW-based blockchain protocols have been defined: *common prefix property* [47, 88], *chain quality property* [47], and *chain growth property* [63]. Intuitively, the chain growth property states that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property aims at indicating the number of honest blocks' contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0, 1)$, the ratio of blocks, that are generated by honest players, in a continuous part of an honest chain is at least μ. We follow the same path to define the security properties for Bitcoin-like PoS-based blockchain protocols, as below.

Definition 44 (Chain growth)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain growth property with parameter* $g \in \mathbb{R}$*, states: for any honest player* P_1 *with local chain* $\mathcal{C}_1$ *at round* r_1*, and honest player* P_2 *with local chain* $\mathcal{C}_2$ *at round* r_2*, where* $P_1, P_2 \in \mathcal{P}$ *and* $r_2 > r_1$*, in the execution* $\mathsf{EXEC}_{\Pi,\mathscr{A},\mathscr{Z}}$*, it holds that* $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g(r_2 - r_1)$.

Definition 45 (Common prefix)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The common prefix property states the following: for any honest player* P_1 *adopting local chain* $\mathcal{C}_1$ *at round* r_1*, and honest player* P *adopting local chain* $\mathcal{C}$ *at round* r*, in the execution* $\mathsf{EXEC}_{\Pi,\mathscr{A},\mathscr{Z}}$*, where* $P_1, P \in \mathcal{P}$ *and* $r \leq r_1$*, it holds that* $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}_1$.

Definition 46 (Chain quality)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain quality property with parameters* μ, ℓ*, where* $\mu \in \mathbb{R}$ *and* $\ell \in \mathbb{N}$*, states: for any honest player* $P \in \mathcal{P}$*, with local chain* $\mathcal{C}$ *in round* r*, in* $\mathsf{EXEC}_{\Pi,\mathscr{A},\mathscr{Z}}$*, it holds, for any* ℓ *consecutive blocks of* $\mathcal{C}$*, the ratio of honest blocks is at least* μ*.*

4.1.3 Unpredictability

The unpredictability property has been investigated by Brown-Cohen et al. [19] in incentive-driven settings. At a high level, predictability means that (certain) protocol players are aware that they will be selected to generate blocks of the blockchain, *before* they actually generate the blocks. (Please see several predictability-based attacks in Supplemental materials 4.11.1.)

In this subsection, we investigate the unpredictability property in the cryptographic setting. For any environment $\mathcal{Z}$ and any adversary $\mathcal{A}$, consider some $\texttt{VIEW}$ in the support of $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. Consider a malicious player $P \in \mathcal{P}$ at round r. Let $\texttt{VIEW}^r$ be the view of all players at round r, and $\mathcal{C}^r$ be the best valid chain of all players in $\texttt{VIEW}^r$. At round r, the adversary $\mathcal{A}$ makes an attempt to predict if the (malicious) player P is able to extend the best chain at a future round r', where $r' > r$. Let $z_P^{r'} \in \{0, 1\}$ be the prediction: here, $z_P^{r'} = 1$ means that the adversary $\mathcal{A}$ predicts that player P can extend the best chain at round r'.

Now we need to introduce another random variable $\bar{z}_P^{r'}$ to indicate if the malicious player P indeed is able to extend the best chain at round r' (as the adversary predicated at an early round r, where $r < r'$) or not. Let $\texttt{VIEW}^{r'}$ be the view of all players at round r', and $\mathcal{C}^{r'}$ be the best valid chain of all players in $\texttt{VIEW}^{r'}$. We set $\bar{z}_P^{r'} := 1$ if there exists a chain $\mathcal{C} = \mathcal{C}^{r'} \| B$ in $\texttt{VIEW}$ with a block B generated by player P at round r', otherwise we set $\bar{z}_P^{r'} := 0$.

We say a prediction $z_P^{r'}$ by the adversary is considered **accurate** if $z_P^{r'} = \bar{z}_P^{r'}$.

Consider a view $\texttt{VIEW}$, and protocol round r, and a malicious player P. For $L \in \mathbb{N}$ and a prediction $z_P^{r'}$ where $r' > r$, we define the predicate $\mathsf{predictable}$ to be true if the prediction $z_P^{r'}$ accurately predicts whether or not player P can generate a new chain at round r' that is L blocks longer than the longest chain at round r. More concretely, we define $\mathsf{predictable}(\texttt{VIEW}, P, L, r, r', z_P^{r'}) := 1$ if and only if the following three conditions hold: *(i)* $r' > r$; *(ii)* $\mathsf{len}(\mathcal{C}^{r'}) + 1 - \mathsf{len}(\mathcal{C}^{r}) = L$; and *(iii)* $z_P^{r'} = \bar{z}_P^{r'}$.

We say a player is L-unpredictable at round r if the adversary cannot predict whether or not the player can generate the next L blocks.

Definition 47 (L-unpredictability)**.** *Consider a blockchain protocol* Π*. For* $L \in \mathbb{N}$*, we say a malicious player* P *is* L*-unpredictable at round* r *if for all* PPT $\mathcal{Z}$*, for all* PPT $\mathcal{A}$*, we have,*

$$\Pr\left[\begin{array}{l|l} \texttt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}; & \mathsf{predictable}(\texttt{VIEW}, P, L, r, r', z_P^{r'}) = 0 \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \texttt{VIEW}^r) & \end{array}\right] > 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

The best possible unpredictability for any PoS protocol is 2-unpredictability. As already shown in their Observation 1 by Brown-Cohen et al. [19], in any PoS protocol, all players can *always* predict whether or not they can generate the next block. In other words, 1-unpredictability cannot be achieved, i.e., 2-unpredictability is the best possible unpredictability. We formally define the best possible unpredictability as follows.

Definition 48 (The best possible unpredictability)**.** *Consider a blockchain protocol* Π*. We say protocol* Π *achieves the best possible unpredictability if for all* PPT $\mathcal{Z}, \mathcal{A}$*,*

for any malicious player P at any round r, we have,

$$\Pr\left[\begin{array}{l|l} \mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}; & \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \mathtt{VIEW}^r) & \mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r', z_P^{r'}) = 0 \end{array}\right] > 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

4.2 An Impossibility Result

We investigate an impossibility result for a group of proof-of-stake protocols, namely, *single-extension PoS protocols*. The result states that a single-extension PoS protocol cannot achieve both the common prefix and the best possible unpredictability properties if honest players control less than 73% of the stake.

We studied existing Bitcoin-like PoS protocols, such as Ouroboros Praos [34] and SnowWhite [32]. These protocols use the same context to generate multiple blocks on the blockchain during an epoch. This means that any two chains within the same epoch share the same context. We refer to this as a *shared-context-extension*. Players attempt to extend the chain by solving puzzles based on the chain's context. If two chains share the same context, players can extend both chains simultaneously. Thus, the shared-context-extension property allows for predictability in the PoS protocol execution. When a player receives one chain with shared-context-extension, they can predict whether or not they can extend the other chain in the future.

In contrast to the existing PoS designs, in Bitcoin, the contexts of two different chains are always different. Therefore, the extension of one chain does not affect the extension of any other chain. This property is referred to as *distinct-context-extension*, and it provides the best possible unpredictability for PoS protocols. Players can only obtain the context of a chain when they receive it, thus allowing them to determine whether they can extend the chain or not.

Based on the distinct-context-extension property, we investigate an impossibility result for single-extension PoS protocols. We first describe the single-extension proof-of-stake framework in Subsection 4.2.1. Then, in Subsection 4.2.2, we state the impossibility result for the single-extension proof-of-stake protocols. In Subsection 4.2.3, we present a new definition of distinct-context-extension property and prove the impossibility result for the single-extension proof-of-stake protocols in Subsection 4.2.4 and Subsection 4.2.5.

4.2.1 Single-extension proof-of-stake protocols

We now describe a design framework, called *single-extension* framework, for Bitcoin-like PoS protocols. Intuitively, in a single-extension protocol, in each round, each honest player identifies only a single "best chain", and then extends the chosen best chain. We remark that, the state-of-the-art PoS protocols (e.g., [32, 34, 7]) can be categorized as single-extension PoS protocols (see Supplemental material 4.11.2 for more details).

We emphasize that, we focus on Bitcoin-like PoS protocols only; the players can generate new blocks in a *non-interactive* fashion by solving PoS puzzles. Our design framework *cannot* be applied for BFT-like protocols (e.g., Algorand [23, 51]); in those protocols, the players must interact with each other to generate new blocks.

Definition 49 (Single-extension framework for PoS protocols)**.** *A single-extension PoS protocol* Π *is executed by a set of player* $\mathcal{P}$*. Initially, each player* $P \in \mathcal{P}$ *holds a key pair* (SK, PK)*. The protocol* Π *is parameterized by deterministic algorithms* $(\mathsf{Context}, \mathsf{Extend}, \mathsf{Validate}, \mathsf{BestChain})$ *as follows:*

- *The validation algorithm* $\mathsf{Validate}$ *takes a chain* $\mathcal{C}$ *and a round* r *as input and returns* 1 *if the chain* $\mathcal{C}$ *is valid at round* r*, and returns* 0 *otherwise.*

– *The context extraction algorithm* Context *takes a valid chain* $\mathcal{C}$ *as input and returns a context* η*. The context* η *is the hash value of some blocks on the chain* $\mathcal{C}$*, based on some hash function* $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^*$*. Note that hash function* hash *will be treated as a random oracle in our analysis.*

More concretely, the input chain $\mathcal{C}$ *is parsed into* $B_0, B_1, \cdots, B_\ell$*, where* $\ell \in \mathbb{N}$*. The algorithm* Context *returns a context* $\eta := \mathsf{hash}(B_{i_1} \| \cdots \| B_{i_t})$*. Here,* $i_j \in [0..\ell]$ *for all* $j \in \{1, \ldots, t\}$*. (Note that, the algorithm* Context *returns* $\perp$ *when the input chain* $\mathcal{C}$ *is invalid.)*

We remark that, the state-of-the-art PoW (e.g., Bitcoin [47, 88]) and PoS (e.g., [32, 34, 7]) use the context extraction algorithms in the above format. For example, in Bitcoin [47, 88], the context is computed as the hash value of the last block on the chain. In Ouroboros Praos [34], the context is computed as the hash value of one or multiple blocks from the previous epoch. In Snow White [32], the context is the concatenation of the random seeds from multiple blocks in the previous epoch; here, the function hash *first truncates the blocks in the previous epoch and obtain the random seeds from those blocks; then it concatenates all the random seeds to obtain the context. More details can be found in the Supplemental material 4.11.2.*

– *The extension algorithm* Extend *is parameterized by a probability* $p \in (0,1)$*. The algorithm* Extend *takes input as a context* η*, a round* r*, and a secret key* SK *and returns a new block* B *or* $\perp$ *(if no new block is generated). Here, the secret key* SK *is generated by a player* P *in the blockchain initialization phase and the corresponding public key of* SK *will be stored in the genesis block. The function* $\mathsf{Extend}(\eta, r, \mathrm{SK})$ *returns a block* B *with probability* p*.*

– *The best chain algorithm* BestChain *takes a set of valid chains* $\mathbb{C}$ *and returns the longest chain* $\mathcal{C}_{\mathrm{best}}$ *as the best chain. Here, the honest player will only* ***extend a***

single chain, *i.e., the longest chain* $\mathcal{C}_{\text{best}}$. *Thus, we name the protocol* ***single-extension***.

The execution of a single-extension protocol consists of two phases as follows.

Blockchain initialization phase. *In this phase, the genesis block will be created; the genesis block consists of a randomness, the public information and the stake distribution of the players. Consider an (initial) group of PoS-players* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *and a security parameter* κ. *Each player* $P_j \in \mathcal{P}$ *generates a pair of public key* PK_j *and private key* SK_j. *The public keys of all players are stored in the genesis block, denoted by* B_0, *of the blockchain system.*

Algorithm 5: A single-extension proof-of-stake protocol Π.

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with key pair (SK, PK) and local chain set $\mathbb{C}$, proceeds as follows.

1 Upon receiving a chain $\mathcal{C}'$, set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$ after verifying $\mathsf{Validate}(\mathcal{C}', r) = 1$;
2 Set $\mathcal{C}_{\text{best}} := \mathsf{BestChain}(\mathbb{C})$;
3 Set $\eta := \mathsf{Context}(\mathcal{C}_{\text{best}})$; Set $B := \mathsf{Extend}(\eta, r, \text{SK})$;
4 **if** $B \neq \perp$ **then**
5 | Set $\mathcal{C} := \mathcal{C}_{\text{best}} \| B$; Add $\mathcal{C}$ to the set $\mathbb{C}$; Broadcast $\mathcal{C}$;

Blockchain extension phase. *A single-extension proof-of-stake protocol* Π *is described in Algorithm 5. In each round* r, *a player* P *with the secret key* SK *proceeds as follows. First, the player* P *computes* $\mathcal{C}_{\text{best}} := \mathsf{BestChain}(\mathbb{C}, r)$. *Here the local set of chains* $\mathbb{C}$ *consists of all valid chains that are received (or generated) by* P. *Then, the player* P *uses the function* Context *to compute the context* η *in the best chain* $\mathcal{C}_{\text{best}}$, *i.e.,* $\eta := \mathsf{Context}(\mathcal{C}_{\text{best}})$. *Finally, based on the context* η, *the current round number* r, *and the secret key* SK, *the player* P *uses the function* Extend *to determine whether or not it can generate a new block. If the player* P *can generate a new block* B, *it creates a new chain* $\mathcal{C} := \mathcal{C}_{\text{best}} \| B$, *adds* $\mathcal{C}$ *to the set of chains* $\mathbb{C}$, *and broadcasts* $\mathcal{C}$ *to all other players.*

4.2.2 Impossibility result for single-extension proof-of-stake protocols

We present an impossibility result for single-extension PoS protocols. More concretely, consider a PoS protocol in the single-extension framework; we can show that, if the PoS protocol achieves the best possible unpredictability, then the protocol cannot simultaneously maintain fundamental security properties, such as the common prefix, when honest players control less than 73% of the stake.

We will prove the impossibility of the result through the distinct-context-extension property. This property is essential for the protocol to achieve the best possible unpredictability. Specifically, it ensures that the contexts of any two different chains in the execution are different, making it impossible for an adversary to predict future chain extensions based on past extensions. However, this property also allows an adversary to amplify their stake by a factor of $e = 2.72$, enabling them to break the common prefix property with only 27% of the stake. Therefore, if the honest players control less than 73% of the stake, it becomes impossible to simultaneously achieve both the distinct-context-extension and common prefix properties, resulting in our impossibility result.

We remark that, the aforementioned impossibility does not hold for single-extension PoW protocols. In these protocols, the property of distinct-context-extension is also necessary for the best possible unpredictability. However, the cost of computing power needed to generate a new block in a PoW protocol is prohibitively high, preventing players from extending multiple chains to improve their chances of generating new blocks. On the other hand, the computing power required to extend a chain in a Proof-of-Stake (PoS) protocol is very cheap, making it possible for an adversary to extend multiple chains and increase their chances of generating new blocks. Therefore, a PoW protocol can achieve the best possible unpredictability and maintain the

security (e.g., common prefix property) when 51% of the mining power is honest.

Let N be the number of players and ρ be the fraction of malicious players in the protocol execution. Let p be the probability that a player can extend a chain in a round. The probability that honest players extend a chain in a round is $\alpha = 1 - (1-p)^{N\cdot(1-\rho)}$. Similarly, the probability that the adversary extends a given chain is $\beta = 1 - (1-p)^{N\cdot\rho}$. Later, we will show that if the protocol Π achieves the distinct-context-extension property, the adversary can amplify its stake by a factor of $e = 2.72$. Therefore, if $\alpha < 2.72\beta$ (i.e., less than 73% of the total stake is honest), the protocol cannot achieve the common prefix property. We are now ready to state the impossibility theorem for protocol Π.

Theorem 50. *Consider a single-extension PoS protocol* Π *that achieves the best possible unpredictability. If* $\alpha < 2.72\beta$*, then protocol* Π *cannot achieve common prefix property.*

We describe in Figure 17, the roadmap for the proof of the impossibility theorem.

Proof. We prove Theorem 50 in the following two steps. First, we show in Lemma 53 that if the single-extension PoS protocol Π achieves the best possible unpredictability, it must achieve *distinct-context-extension* property. Secondly, we show in Lemma 59 that if the single-extension PoS protocol Π achieves distinct-context-extension property, it cannot achieve common prefix property if $\alpha < 2.72\beta$. Specifically, if protocol Π achieves distinct-context-extension property, then the adversary can amplify its stake by a factor $e = 2.72$ by extending all valid chains. Thus, if $\alpha < 2.72\beta$, the adversary can extend the chain faster than the honest player. Hence, they can break the common prefix property by keep its chain hidden for a sufficient long period and then publish the hidden chain. As the chain of the adversary is longer, it will be come the new best chain. Since the new best chain does not share a common prefix

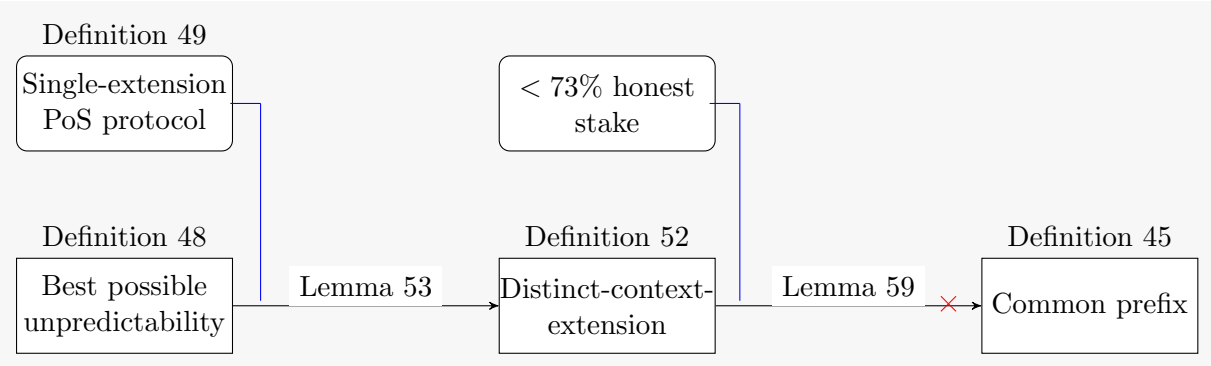


Figure 17: The roadmap for the proof of our impossibility result (Theorem 50). First, we show in Lemma 53 that if a single-extension PoS protocol achieves the best possible unpredictability, it must achieve distinct-context-extension property. Secondly, we show in Lemma 59 that if a single-extension PoS protocol achieves distinct-context-extension property and the honest players control less than 73% of stake, the protocol *cannot* achieve common prefix property.

with the old best chain of the honest players, the common prefix property does not hold. □

We remark that we are the first to show the impossibility result for single-extension PoS protocols. To show the impossibility, we need to formally define the single-extension PoS protocol and introduce a new concept of a distinct-context-extension property. Then, based on the distinct-context-extension property, we can prove the impossibility result in two steps, as mentioned above. The proof of the second step has been shown in [7]. However, the impossibility result has not been presented in those works.

4.2.3 Distinct-context-extension

We introduce a new definition for the *distinct-context-extension* property. As previously stated, we will use this property to prove an impossibility result for single-extension PoS protocols. The *distinct-context-extension* property states that the contexts of any chains in the execution must be distinct. This prevents the adversary from predicting the extension of a chain in the future based on an existing chain in the past. Hence, the distinct-context-extension property is needed in order to achieve the best possible unpredictability. At the same time, this also allows the adversary to amplify their stake by a factor of $e = 2.72$, enabling them to extend the chain faster than the honest players and break the common prefix property if they control more than 27% of the stake. Therefore, it is impossible for a single-extension protocol to simultaneously achieve the distinct-context-extension and common prefix properties if the adversary controls more than 27% of the stake.

Additionally, we define *shared-context-extension*, which is the counterpart of distinct-context-extension. We say two chains in the execution are shared-context-extension if the contexts of the two chains are the same. The shared-context-extension allows the adversary to predict whether or not it can extend a future chain that has not yet been generated. Specifically, if the future chain shares the same context as the current chain, the adversary can base its prediction (for the future chain) on the extension of the current chain. We note that existing protocols, such as Ouroboros Praos [34] and SnowWhite [32], have the shared-context-extension property, while our protocol in Section 4.3 achieves the distinct-context-extension property.

We now present the definition of distinct-context-extension and shared-context-extension for two chains. The toy examples of distinct-context-extension and shared-context-extension can be seen in Figure 18 and Figure 19, respectively.

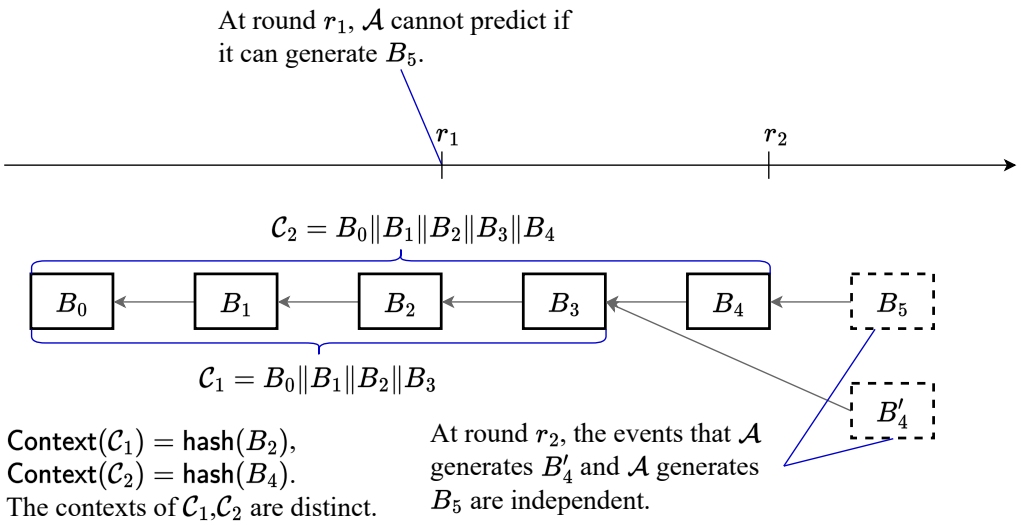


Figure 18: A toy example of distinct-context-extension for two chains. Consider two chains $\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$ and $\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4\|B_5$. Here, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_5)$. As $B_3 \neq B_5$, we have, $\mathsf{Context}(\mathcal{C}_1) \neq \mathsf{Context}(\mathcal{C}_2)$. In other words, $\mathcal{C}_1$ and $\mathcal{C}_2$ are distinct-context-extension. At round r_2, the events that the adversary $\mathcal{A}$ can extend the chain $\mathcal{C}_1$ to generate a new block B'_4 and the probability that $\mathcal{A}$ can extend the chain $\mathcal{C}_2$ to generate a new block B_6 are independent. At round r_1, the adversary $\mathcal{A}$ has not yet received the chain $\mathcal{C}_2$. Therefore, it cannot predict whether it can extend $\mathcal{C}_2$ to generate block B_6.

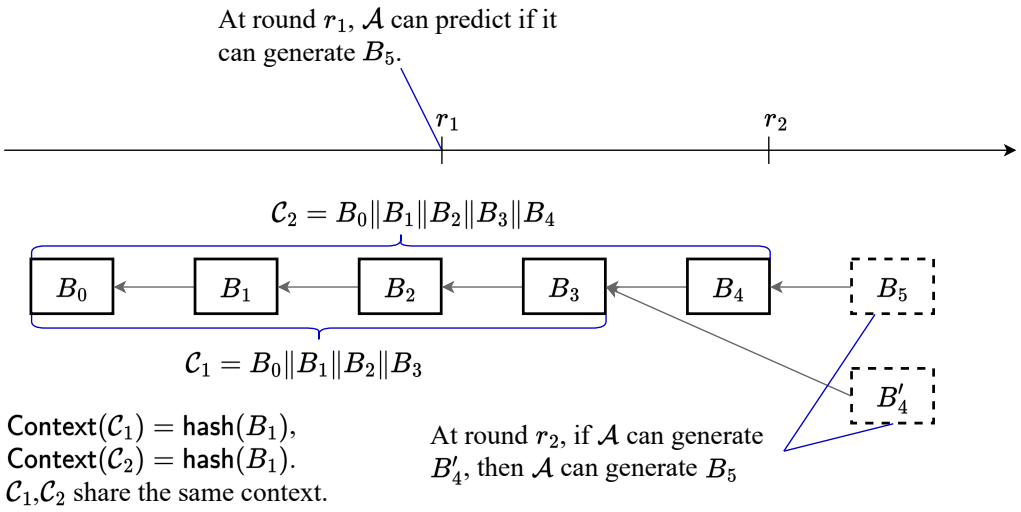


Figure 19: A toy example of shared-context-extension for two chains. The definition of function $\mathsf{Context}$ here is very different from that in Figure 18. Consider two chains $\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$ and $\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4\|B_5$. We stress that, here, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_1)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_1)$. While in Figure 18, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_5)$. We have, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{Context}(\mathcal{C}_2)$. In other words, $\mathcal{C}_1$ and $\mathcal{C}_2$ are shared-context-extension. At round r_2, if the adversary $\mathcal{A}$ can extend the chain $\mathcal{C}_1$ to generate a new block B'_4, it can also extend the chain $\mathcal{C}_2$ to generate a new block B_6. Thus, at round r_1, when the adversary $\mathcal{A}$ has received $\mathcal{C}_1$ but not yet received $\mathcal{C}_2$, the adversary can predict whether or not it can extend $\mathcal{C}_2$ to generate block B_6 at round r_2.

Definition 51 (Distinct and shared-context-extension for two chains)**.** *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* Validate, BestChain, Context, *and* Extend. *Let* $\mathcal{P}$ *be the set of players. For any adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Consider two chains* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *in* VIEW*.*

- *We say the extensions of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* distinct-context-extension *if the contexts of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are distinct, i.e.,* $\mathsf{Context}(\mathcal{C}_1) \neq \mathsf{Context}(\mathcal{C}_2)$*. We write* $\mathsf{distinct\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. In this case, the event that the adversary* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_1$ *and the event that* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_2$ *are independent.*
- *We also consider the flip side of distinct-context-extension, namely, shared-context-extension. We say the extensions of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* shared-context-extension *if the contexts of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are the same, i.e.,* $\mathsf{Context}(\mathcal{C}_1) = \mathsf{Context}(\mathcal{C}_2)$*. We write* $\mathsf{shared\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. In this case, if the adversary* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_1$*, it can also extend the chain* $\mathcal{C}_2$*, and vice versa.*

We are now ready to define the distinct-context-extension property for a PoS protocol. Intuitively, we say that a protocol achieves the distinct-context-extension property if all different chains in the protocol's execution have distinct contexts.

Definition 52 (Distinct-context-extension for all chains in the executions)**.** *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* Validate, BestChain, Context, *and* Extend. *Let* $\mathcal{P}$ *be the set of players. For any adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Consider some round* r*; let* $\mathbb{C}^r$ *be the set of all chains that appear on the view of some players (or the adversary) in* $\mathtt{VIEW}^r$*. Here,* $\mathtt{VIEW}^r$ *is the prefix of* VIEW *up until round* r*. We overload the predicate* distinct-context-extension *for a view* VIEW*. Intuitively,*

a view $\texttt{VIEW}$ *is distinct-context-extension if all different chains in* $\texttt{VIEW}$ *have distinct contexts. More concretely, we say a view* $\texttt{VIEW}$ *is distinct-context-extension if and only if for any round* r*, for any chains* $\mathcal{C}_1, \mathcal{C}_2 \in \mathbb{C}^r$ *such that* $\mathcal{C}_1 \neq \mathcal{C}_2$*, we have,* $\mathsf{distinct\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. We write* $\mathsf{distinct\text{-}context\text{-}extension}(\texttt{VIEW}) = 1$*. We say protocol* Π *achieves* distinct-context-extension property *if for every* PPT $\mathcal{Z}, \mathcal{A}$*, we have,*

$$\Pr[\texttt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \mid \mathsf{distinct\text{-}context\text{-}extension}(\texttt{VIEW}) = 1] = 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

4.2.4 Achieving the best possible unpredictability via distinct-context-extension

We prove that a single-extension PoS protocol can only achieve the best possible unpredictability if it satisfies the distinct-context-extension property. Intuitively, as shown in Figure 19, if there are two chains that are shared-context-extension, the adversary can predict whether or not it can extend a chain in the future (i.e., the chain is not yet generated). This contradicts the best possible unpredictability. More concretely, consider two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ that share a context extension. Without loss of generality, we assume that the length of $\mathcal{C}_1$ is greater than the length of $\mathcal{C}_2$. We define $\mathcal{C}$ as the longest common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. Recall that, the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ are computed using a hash function (which will be treated as a random oracle) over some blocks on the two chains. Since the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ are the same, the shared context must be extracted from the longest common prefix $\mathcal{C}$ of $\mathcal{C}_1$ and $\mathcal{C}_2$. Upon receiving the chain $\mathcal{C}$ at round r, player P can predict whether or not they can extend $\mathcal{C}_1$. As the length of $\mathcal{C}_1$ is greater than the length of $\mathcal{C}$, player P is 2-predictable at round r, which contradicts the best possible unpredictability.

Lemma 53. *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* $\mathsf{Validate}$, $\mathsf{BestChain}$, $\mathsf{Context}$, *and* Extend. *If the protocol* Π *achieves the best possible unpredictability, then it achieves distinct-context-extension property.*

Proof. We assume toward contradiction that there exist two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \neq \mathcal{C}_2$ and the extensions of those two chains are shared-context-extension. We will prove that the protocol Π cannot achieve the best possible unpredictability, i.e., there exists a round r and a malicious player P such that the player P is 2-predictable at round r.

Let $\eta_1 = \mathsf{Context}(\mathcal{C}_1)$ and $\eta_2 = \mathsf{Context}(\mathcal{C}_2)$. We have, $\eta_1 = \eta_2$. We parse $\mathcal{C}_1$ into $B_0 \| B_1 \| \cdots \| B_{\ell_1}$ and parse $\mathcal{C}_2$ into $B'_0 \| B'_1 \| \cdots \| B'_{\ell_2}$. Here, $\ell_1 = \mathsf{len}(\mathcal{C}_1)$ and $\ell_2 = \mathsf{len}(\mathcal{C}_2)$ are the lengths of $\mathcal{C}_1, \mathcal{C}_2$, respectively. Without loss of generality, we assume $\ell_1 \geq \ell_2$. Let r_1, r_2 be the round where $\mathcal{C}_1$, $\mathcal{C}_2$ are generated. Here, $\mathcal{C}_1$ is the best chain at round r_1.

Let $\mathcal{C}$ be the longest common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. Let $\ell = \mathsf{len}(\mathcal{C})$ be the length of chain $\mathcal{C}$. We have, for all $i \in [0..\ell]$, $B_i = B'_i$; and for all $i \in [\ell + 1, \ell_2]$, $B_i \neq B'_i$. Let r be the round where $\mathcal{C}$ is generated. We will prove that player P is 2-predictable at round r. In other words, at round r, the player P can predict weather or not she/he can extend the chain $\mathcal{C}_1$ at round r.

We will show that the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ can be computed when the chain $\mathcal{C}$ is generated. As shown in Definition 49, the contexts are computed based on the hash values of some blocks on the chain. Let $B_{i_1}, \cdots, B_{i_t}$ be the blocks that are used to compute the context of $\mathcal{C}_1$, where $t \in \mathbb{N}$ and for all $j \in [t]$, $i_j \in [0..\ell_1]$. We have, $\eta_1 = \mathsf{hash}(B_{i_1} \| \cdots \| B_{i_t})$. Let $B'_{i'_1}, \cdots, B'_{i'_{t'}}$ be the blocks that are used to compute the context of $\mathcal{C}_2$, where $t \in \mathbb{N}$ and for all $j \in [t']$, $i'_j \in [0..\ell_2]$. We have,

$\eta_2 = \mathsf{hash}(B'_{i'_1} \| \cdots \| B'_{i'_{t'}})$.

The function hash is treated as a random oracle[2]. Note that $\eta_1 = \eta_2$. Now we have, $B_{i_1} \| \cdots \| B_{i_t} = B'_{i'_1} \| \cdots \| B'_{i'_{t'}}$. Thus, we have $t = t'$ and for all $j \in [t]$, $B_{i_j} = B'_{i'_1}$. Hence, all blocks $B_{i_1}, \cdots, B_{i_t}$ must belong to the chain $\mathcal{C}$, the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. In other words, the contexts η_1 and η_2 must be obtained from the chain $\mathcal{C}$.

Next, we will show that chain $\mathcal{C}_1$ is longer than chain $\mathcal{C}$, i.e., $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C})$. We consider two cases based on the length of $\mathcal{C}_1$ and $\mathcal{C}_2$ as follows:

- Case 1: $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)$. As $\mathcal{C}$ is a prefix of $\mathcal{C}_2$, we have, $\mathsf{len}(\mathcal{C}_2) \geq \mathsf{len}(\mathcal{C})$. Thus $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2) \geq \mathsf{len}(\mathcal{C})$.
- Case 2: $\mathsf{len}(\mathcal{C}_1) = \mathsf{len}(\mathcal{C}_2)$. Assume toward contradiction that $\mathsf{len}(\mathcal{C}_1) = \mathsf{len}(\mathcal{C})$. As $\mathcal{C}$ is a prefix of $\mathcal{C}_1$, we have $\mathcal{C} = \mathcal{C}_1$. Similarly, we have $\mathcal{C} = \mathcal{C}_2$. Thus, we have $\mathcal{C}_1 = \mathcal{C}_2$ (this contradicts the condition that $\mathcal{C}_1 \neq \mathcal{C}_2$). Hence, we have, $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C})$.

Let r be the round where player P receives $\mathcal{C}$. At round r, player P can calculate the context η_1 of the chain $\mathcal{C}_1$. At round r, the adversary makes a prediction $z_P^{r_1}$, where

$$z_P^{r_1} = \begin{cases} 0, \text{ if } \mathsf{Extend}(\eta_1, r_1, \mathrm{SK}) = \perp, \\ 1, \text{ if } \mathsf{Extend}(\eta_1, r_1, \mathrm{SK}) \neq \perp . \end{cases}$$

If the function $\mathsf{Extend}(\eta_1, r_1, \mathrm{SK})$ returns a block, i.e., $\mathsf{Extend}(\eta_1, r_1, \mathrm{SK}) \neq \perp$, player P can extend the chain $\mathcal{C}_1$ at round r_1. Hence, the prediction $z_P^{r_1} = 1$ is always accurate. Furthermore, since $\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq 1$, i.e., $\mathsf{len}(\mathcal{C}_1) + 1 - \mathsf{len}(\mathcal{C}) \geq 2$, we have, $\mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r_1, z_P^{r_1}) = 1$. In other words, the malicious player P is 2-predictable at round r. This contradicts the fact that protocol Π achieves the best

[2]It is sufficient to assume that hash is collision resistant hash function in this proof.

possible unpredictability, i.e., player P must be 2-unpredictable at every round.

□

We note that the single-extension protocols in [32, 34] cannot achieve the best possible unpredictability. The execution of these protocols is divided into epochs, each consisting of $O(\kappa)$ blocks. In these protocols, the context is computed based on the hash values of the blocks in the previous epoch. As a result, all chains in the same epoch share the same context. Thus, at the beginning of each epoch, malicious players can predict whether or not they can extend their chains in the current epoch. Bagaria et al. [7] proposed a single-extension protocol with a constant-sized epoch. In this protocol, the context of a chain is computed as the hash value of the last block in the previous epoch. If each epoch consists of at least two blocks, the protocol cannot achieve the best possible unpredictability. This is because all the chains in the same epoch share the same context. On the other hand, if each epoch consists of only one block, the protocol can achieve the best possible unpredictability. Now, the protocol achieves distinct-context-extension property. Jumping ahead, in Subsection 4.2.5, we will show that the protocol requires 73% of honest stake to achieve the security properties.

4.2.5 Breaking the common prefix property via distinct-context-extension

We show that if a single-extension proof-of-stake (PoS) protocol achieves the distinct-context-extension property, the adversary can violate the common prefix property if the honest players control less than 73% of the stake. More specifically, based on the distinct-context-extension property, the adversary can amplify their stake by at least a factor of $e = 2.72$. Therefore, if the honest players control less than 73% of the stake, the adversary can extend chains faster than the honest players

and thus break the common prefix property of the protocol.

The chain growth of the adversary. We establish a bound on the chain growth of the adversary as follows. We demonstrate that for any valid chain $\mathcal{C}$ at any round r, there exists an adversary who can extend the chain C with a probability of at least β. Given that the protocol achieves the distinct-context-extension property, the extensions of all chains are independent. We model the chain extension of the adversary as a random tree, where each branch of the tree represents a chain in the block tree and the extensions of the branches are modeled as independent random variables.

Claim 54. *Adversarial extension Consider any proof-of-stake protocol* Π *with a set of player* P*. For some adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* $\texttt{VIEW}$ *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Let* $\mathbb{C}^r$ *be the set of chains in the view of all players at round* r*. At round* r*, the adversary makes an attempt to extend a chain* $\mathcal{C} \in \mathbb{C}^r$*. Specifically, the adversary* $\mathcal{A}$*, takes inputs as a chain* $\mathcal{C} \in \mathbb{C}^r$*, the round number* r*, and output a block* B*. For every,* PPT $\mathcal{Z}$*, there exists an adversary* $\mathcal{A}$ *such that, at any round* r*, we have,*

$$\Pr\left[\begin{array}{l|l} \texttt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}; & \\ \mathcal{C} \leftarrow \mathbb{C}^r; & \mathsf{Validate}(\mathcal{C}\|B, r) = 1 \\ B \leftarrow \mathcal{A}(\mathcal{C}, r); & \end{array}\right] \geq \beta.$$

Proof. Consider an adversary $\mathcal{A}$ that corrupts $N \cdot \rho$ players at the start of the protocol. The adversary $\mathcal{A}$ extends all chains in $\mathbb{C}^r$ for each round r. For each malicious player P, let SK be their secret key. The adversary instructs player P to run $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \text{SK})$. Recall that, the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \text{SK})$ returns $\perp$ if no new block is generated. Otherwise, if $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \text{SK}) \neq \perp$, i.e., the algorithm return a new block, the player P can generate a new block.

The adversary can generate a new block in a round r if there exists a malicious

player P' successfully generates a new block, i.e., $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \mathrm{SK}') \neq \perp$, where SK' is the secret key of P'. In this case, the adversary returns the block that is output by the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \mathrm{SK}')$. Otherwise, if all malicious players cannot generate a new block at round r, the adversary returns $\perp$.

Recall that, the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \mathrm{SK})$ returns a new block with probability p. As the number of malicious players is $N \cdot \rho$, the probability that there exists an malicious play can extend the chain $\mathcal{C}$ at round r is $1 - (1-p)^{N \cdot \rho} = \beta$. In this case, the adversary returns a block core that is generated by a malicious player at round r. □

Next, we show that the adversary can amplify its stake by a factor of $e = 2.72$. To do this, we consider an adversary that extends all chains. The extension of the adversary is modeled as a random tree, in which each branch represents the extension of a chain. Based on Lemma 54, the probability of the adversary extending a chain in each round is at least β. Additionally, as stated in Lemma 53, to achieve the best possible unpredictability, the extensions of all chains must be distinct-context-extension. Thus, the probabilities of the adversary extending the chains are independent. The chain extension of the adversary is modeled as a random tree with independent extensions in each branch. To bound the growth rate of the chain, we first bound the number of branches in the random tree and then, based on the number of branches and the growth rate of each branch, we can determine the maximum length of all branches in the random tree.

Before presenting the detailed proofs, we introduce a useful inequality as follows.

Claim 55 (Theorem 1 in [22])**.** *Consider a Poisson random variable X that has the expected value of λ. We have the following inequalities.*

- *For any $\epsilon > 0$, we have,* $\Pr[X > \lambda \cdot (1+\epsilon)] \leq e^{-\lambda \frac{\epsilon^2}{2(1+\epsilon)}}$.

- *For any* $0 < \epsilon < 1$*, we have,* $\Pr[X < \lambda \cdot (1-\epsilon)] \leq e^{-\lambda \frac{\epsilon^2}{2(1+\epsilon)}}$.

Claim 56 (Theorem 3 in [52])**.** *Let* $X_1, X_2, \cdots, X_t$ *be identical independent random variables in range* $[0,1]$ *with an expected value of* λ*. Then, for any* $\epsilon > 0$*, we have* $\Pr[\sum_{i=1}^{t} X_i < (1-\epsilon) \cdot t \cdot \lambda] \leq e^{-\Omega(t)}$.

We describe the adversary's chain extension as a branching process, as follows. Let Z_t be the set of all branches at round t, where $t \in \mathbb{N}$, and G_t be the number of branches in Z_t. At the beginning, there is only one branch of length 0, i.e., $Z_0 = \{0\}$ and $G_0 = 1$. Let X be a Poisson random variable with an expected value of β. Let $X_{t,i}$ be the random variable that represents the random process in the i-th branch in Z_t. Here, $X_{t,i}$ are independent and identically distributed random variables of X. Let $\ell_{t,i}$ be the length of the i-th branch in Z_t. We will add $X_{t,i}+1$ branches with the length $\ell_{t,i}, \ell_{t,i}+1, \cdots, \ell_{t,i}+X_{t,i}$ into Z_{t+1}. We denote T_t as the maximum length of all branches in Z_t, i.e., $T_t = \max_{i \in \{1,2,\cdots,G_t\}} \ell_{t,i}$. The maximum length T_t is equivalent to the length of the longest chain. To bound the adversary's chain growth, we first bound the number of different branches at the end of the process (see Lemma 57). Then, we use the union bound for the maximum length of all branches.

Claim 57. *Consider the set of branches* Z_t *at time* t*. For any* $\epsilon'' > 0$*, we have* $\Pr[G_t < (\beta+1)^{(1-\epsilon'') \cdot t}] < e^{-\Omega(t)}$.

Proof. In each round, on average, the adversary can create $\beta+1$ new branches from a branch in the previous rounds. Thus, for $j \in \mathbb{N}$, we have $\mathbb{E}\big[\frac{G_{j+1}}{G_j}\big] = \beta+1$. In other words, we have $\mathbb{E}\big[\log(G_{j+1}) - \log(G_j)\big] = \log(\beta+1)$.

Let $Q_1, Q_2, \cdots, Q_t$ be independent and identically distributed random variables

with the expected value of $\log(\beta+1)$. We have, $\log G_t = \sum_{j=1}^{t} Q_j$. Therefore,

$$\begin{aligned}&\Pr\left[G_t < (\beta+1)^{(1-\epsilon'')\cdot t}\right] = \Pr\left[\log(G_t) < (1-\epsilon'')\cdot t\cdot\log(\beta+1)\right]\\ =&\Pr\left[\sum_{j=1}^{t} Q_j < (1-\epsilon'')\cdot t\cdot\log(\beta+1)\right] < e^{-\Omega(t)}.\end{aligned}$$

□

Claim 58. *Consider a single-extension proof-of-stake protocol* Π *satisfies distinct-context-extension property. For some adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* $\texttt{VIEW}$ *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Let* $\mathbb{C}^{r_1}$ *be the set of chains at round* r_1*. The adversary takes inputs as a chain* $\mathcal{C}_1 \in \mathbb{C}^{r_1}$ *and the round numbers* r_1, r_2 *and outputs a chain* $\mathcal{C}_2$*. For every,* PPT $\mathcal{Z}$*, there exists an adversary* $\mathcal{A}$ *such that, at any round* r_1, r_2*, where* $r_2 - r_1 = t$ *and* $t = \Omega(\kappa)$*, we have,*

$$\Pr\left[\begin{array}{l|l}\texttt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}};\ \mathcal{C}_1 \leftarrow \mathbb{C}^{r_1}; & \big(\mathsf{Validate}(\mathcal{C}_2, r_2) = 1\big)\\ \mathcal{C}_2 \leftarrow \mathcal{A}(\mathcal{C}_1, r_1, r_2); & \bigwedge\ \big(\mathcal{C}_1 \preceq \mathcal{C}_2\big)\\ \ell_1 := \mathsf{len}(\mathcal{C}_1); \ell_2 := \mathsf{len}(\mathcal{C}_2); & \bigwedge\ \big(\ell_2 - \ell_1 > (1-\epsilon)\cdot e\cdot\beta\cdot t\big)\end{array}\right] \geq 1 - e^{-\Omega(\kappa)},$$

where $e = 2.72$*.*

Proof. Let $Y := \sum_{j=1}^{t} X_j$ be a Poisson random variable with the expected value of $k\cdot\beta$. We have,

$$\begin{aligned}\Pr\left[T_t < (1-\epsilon)\cdot t\cdot\beta\cdot e\right] \leq& \sum_{i\in\{1,2,\cdots,G_t\}} \Pr\left[\ell_{t,i} < (1-\epsilon)\cdot t\cdot\beta\cdot e\right]\\ \leq& G_t\cdot\Pr\left[Y < (1-\epsilon)\cdot t\cdot\beta\cdot e\right]\\ \leq& (\beta+1)^{(1+\epsilon'')\cdot t}\cdot\Pr\left[Y < (1-\epsilon)\cdot t\cdot\beta\cdot e\right] + e^{-\Omega(\kappa)}\\ \leq& \Pr\left[Y < (1-\epsilon')\cdot t\cdot\beta\right] + e^{-\Omega(\kappa)} = e^{-\Omega(\kappa)}.\end{aligned}$$

□

Breaking common prefix. Since the adversary can amplify its stake by a factor $e = 2.72$, the adversary can extend the chain faster than the honest players it control more than 27% of stake. Thus, the adversary can keep its blocks hidden and then publishes those block when the length of hidden chain is bigger than κ. Now, the hidden chain will become the new best chain and it does not share a common prefix with the previous best chain.

Lemma 59. *Assume $\alpha < e \cdot \beta$, where $e = 2.72$. Consider a single-extension proof-of-stake protocol Π that is parameterized by four algorithms:* $\mathsf{Validate}$, $\mathsf{BestChain}$, $\mathsf{Context}$, *and* Extend. *If protocol Π achieves distinct-context-extension property, then it cannot achieve common prefix property.*

Proof. Consider an adversary that extends a set of chains and keep them private from the beginning of the protocol execution. Consider a round $r = \Omega(\kappa)$, let $\mathcal{C}_1$ be the best chain that is generated by the adversary. Here, $\mathcal{C}_1$ is private, i.e., it is hidden from the honest players. From Claim 58, we have $\Pr[\mathsf{len}(\mathcal{C}_1) < (1-\epsilon_1) \cdot r \cdot e \cdot \beta] < e^{-\Omega(\kappa)}$, where $e = 2.72$. Let $\mathcal{C}_2$ be the best public chain at round r. Using Chernoff bound, we have, $\Pr[\mathsf{len}(\mathcal{C}_2) > (1+\epsilon_2) \cdot r \cdot \alpha] < e^{-\Omega(\kappa)}$. Here, we choose ϵ_1, ϵ_2 such that $(1-\epsilon_1) \cdot e \cdot \beta > (1+\epsilon_2) \cdot \alpha$. We have, $\Pr[\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)] > \Pr[(1-\epsilon_1) \cdot r \cdot e \cdot \beta > (1+\epsilon_2) \cdot \alpha] - e^{-\Omega(\kappa)} = 1 - e^{-\Omega(\kappa)}$.

At some round $r' = \Omega(\kappa)$ such that $\mathsf{len}(\mathcal{C}_1) > \kappa$, the adversary publishes the best private chain $\mathcal{C}_1$. Recall that, $\Pr[\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)] > 1 - e^{-\Omega(\kappa)}$. In other words, with overwhelming probability, the private chain $\mathcal{C}_1$ is longer than the best public chain $\mathcal{C}_2$. Therefore, the honest players will adopt $\mathcal{C}_1$ as the best public chain.

Note that, all blocks in the private chains $\mathcal{C}_1$ (except the genesis block) do not belong to the public chain $\mathcal{C}_2$. Thus, we have, $\mathcal{C}_1[\neg\kappa] \not\preceq \mathcal{C}_2$. Therefore, the common prefix property does not hold. □

4.3 Greedy Strategies: How to overcome the impossibility

In Section 4.2, we have demonstrated the impossibility of single-extension PoS protocols. Specifically, we have shown that a single-extension PoS protocol that achieves the best possible unpredictability cannot simultaneously achieve the common prefix property if honest players control less than 73% of the stake. In this section, we introduce a *multi-extension* framework that allows honest players to extend multiple chains. We then present greedy strategies that follow this framework. In these strategies, honest players are allowed to extend multiple chains that are "close" to each other. Additionally, we design a new tiebreak rule for the multi-extension protocol to maximize the chain growth of honest players. Our multi-extension protocol can achieve the best possible unpredictability while only requiring a much smaller fraction (e.g., 57%) of the honest stake to achieve the security properties.

For simplicity, we consider the idealized "flat" model where all PoS-players have the same number of stake and register exactly one public key in the genesis block. In a non-flat model where the PoS-players may have different numbers of stake, we can set the difficulty of the hash inequality based on the number of stakes that the player controls (see more details in Section 4.9.2). Plus, we assume all protocol players have their stake registered at the beginning of the protocol execution. In Section 4.9.3, we will turn to consider a dynamic stake distribution and use a similar strategy as in [6] to allow new players to join the system during the protocol execution.

4.3.1 Multi-extension proof-of-stake protocols

We say a PoS protocol is a multi-extension protocol if in each round, each honest player is allowed to extend multiple chains. By extending multiple chains, the honest players can extend the best chain faster, compared to the single-extension protocol.

Definition 60 (Multi-extension framework for PoS protocols)**.** *A multi-extension PoS protocol* Π° *is parameterized by 4 deterministic algorithms* ($\mathsf{Validate}^{\circ}$, $\mathsf{BestChainSet}^{\circ}$, $\mathsf{Context}^{\circ}, \mathsf{Extend}^{\circ}$) *as follows:*

- *The validation algorithm* $\mathsf{Validate}^{\circ}$ *takes a chain* $\mathcal{C}$ *and a round* r *as input and returns* 1 *if the chain* $\mathcal{C}$ *is valid at round* r*, and returns* 0 *otherwise.*
- *The context extraction algorithm* $\mathsf{Context}^{\circ}$ *takes a valid chain* $\mathcal{C}$ *as input and returns a context* η*. If the input chain is invalid, the algorithm returns* $\perp$*.*
- *The extension algorithm* Extend° *is parameterized by a probability* $p \in (0, 1)$*. The algorithm takes input as a context* η*, a round* r*, and a secret key* SK*, and returns a new block* B *or* $\perp$ *(if no new block is generated). Here, the secret key* SK *is generated by a player* P *in the blockchain initialization phase, and the corresponding public key of* SK *will be stored in the genesis block. The function* $\mathsf{Extend}^{\circ}(\eta, r, \mathrm{SK})$ *returns a block* B *with probability* p*.*
- *The best chain set algorithm* $\mathsf{BestChainSet}^{\circ}$ *takes a set of chains* $\mathbb{C}$ *and returns a set of the best chains* $\mathbb{C}_{\mathrm{best}}$*. Here, the honest players will* ***extend multiple chains****, i.e., all the chains in the set of the best chains* $\mathbb{C}_{\mathrm{best}}$*. Thus, we name the protocol* ***multi-extension****.*

We note that the descriptions of algorithms $\mathsf{Validate}^{\circ}$*,* $\mathsf{Context}^{\circ}$*, and* Extend° *are identical to the descriptions of algorithms* $\mathsf{Validate}$*,* $\mathsf{Context}$*, and* Extend *in the single-extension PoS protocol, defined in Definition 49. The difference between the single-extension and multi-extension protocols is the utilization of different algorithms, namely* $\mathsf{BestChain}$ *and* $\mathsf{BestChainSet}^{\circ}$*, for determining the chains to be extended. The single-extension protocol uses the best chain algorithm* $\mathsf{BestChain}$ *to select a single best chain. Meanwhile, the multi-extension protocol employs the best chain set algorithm*

$\mathsf{BestChainSet}^{\circ}$, *which returns a set of multiple best chains. The advantage of extending multiple chains is that honest players can extend the best chain faster compared to the single-extension protocol.*

Blockchain initialization phase. *In this phase, the genesis block will be created; the genesis block consists of a randomness, the public information and the stake distribution of the players. Consider an (initial) group of PoS-players* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *and a security parameter* κ. *Each player* $P_j \in \mathcal{P}$ *generates a pair of public key* PK_j *and private key* SK_j. *The public keys of all players are stored in the genesis block of the blockchain system. We let* B_0 *denote the genesis block. .*

Algorithm 6: A multi-extension proof-of-stake protocol Π°.

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with key pair $(\mathrm{SK}, \mathrm{PK})$ and local chain set $\mathbb{C}$, proceeds as follows.

Upon receiving a chain $\mathcal{C}'$, verify $\mathsf{Validate}^{\circ}(\mathcal{C}', r) = 1$ and set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$;
Set $\mathbb{C}_{\mathrm{best}} := \mathsf{BestChainSet}^{\circ}(\mathbb{C})$;
for $\mathcal{C} \in \mathbb{C}_{\mathrm{best}}$ **do**
  $\eta := \mathsf{Context}^{\circ}(\mathcal{C})$; $B := \mathsf{Extend}^{\circ}(\eta, r, \mathrm{SK})$;
  if $B \neq \perp$ **then**
    $\mathcal{C}' := \mathcal{C} \| B$; Add $\mathcal{C}'$ to $\mathbb{C}$; Broadcast $\mathcal{C}'$;

Blockchain extension phase. *A multi-extension proof-of-stake protocol* Π *is described in Algorithm 6. In each round* r, *a player* P *with the secret key* SK *proceeds as follows. First, the player* P *set* $\mathbb{C}_{\mathrm{best}} := \mathsf{BestChainSet}^{\circ}(\mathbb{C})$. *Here the local set of chains* $\mathbb{C}$ *consists of all valid chains that are received (or generated) by* P. *Then, for each chain* $\mathcal{C} \in \mathbb{C}_{\mathrm{best}}$, *the player* P *uses the function* $\mathsf{Context}^{\circ}$ *to extract the context* η *in the best chain* $\mathcal{C}$, *i.e.,* $\eta := \mathsf{Context}^{\circ}(\mathcal{C}, r)$. *Finally, based on the context* η, *the current round number* r, *and the secret key* SK, *the player* P *uses the function* Extend° *to determine whether or not it can generate a new block. If the player* P *can generate a new block* B, *it creates a new chain* $\mathcal{C}' := \mathcal{C} \| B$, *adds* $\mathcal{C}'$ *to the set of chains* $\mathbb{C}$ *and*

broadcasts $\mathcal{C}'$ to all other players.

Remark 1. *We remark that, there are other ways to design multi-extension protocols. However, to simplify the presentation, we focus on the design in Definition 60. We used this design of multi-extension protocols for our protocol in Section 4.3.*

4.3.2 Greedy strategies

We allow the PoS players to take a *greedy* strategy to extend the chains in a protocol execution: instead of extending a single best chain (i.e., the longest chain), the players are allowed to extend *a set of best chains*, expecting to extend the best chain faster. This is possible because extending the chains in a PoS protocol is "very cheap". We remark that, the set of best chains should be carefully chosen; otherwise, the protocol may not be secure. In our greedy strategy, the honest player extends the set of chains that share the same common prefix after removing the last few blocks. With this strategy, the security of the protocol is guaranteed. Next, we will formally study the greedy strategies.

Distance-greedy strategies. We first introduce *distance-greedy strategies* for honest protocol players. Consider a blockchain protocol execution. In each player's local view, there are multiple chains, which can be viewed as a tree. More concretely, the genesis block is the root of the tree, and each path from the root to another node is essentially a chain. The tree will "grow": the length of each existing chain may increase, and new chains may be created, round after round. Before giving the formal definition for distance-greedy strategies, we define the "distance" between two chains in a tree. Intuitively, we say the distance from a "branch" chain to a "reference" chain is d if we can obtain a prefix of the reference chain by removing the last d blocks of the branch chain.

Definition 61 (Distance between two chains)**.** *Let $\mathcal{C}$ be a chain of length ℓ. We view $\mathcal{C}$ as the "reference" chain, and now consider $\mathcal{C}_1$ to be a "branch" chain (of the reference chain). Let ℓ_1 be the length of $\mathcal{C}_1$. Next, we define the distance between the reference chain $\mathcal{C}$ and the branch chain $\mathcal{C}_1$, and we use* $\mathsf{distance}$*(branch chain $\rightarrow$ reference chain), i.e.,* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C})$ *to denote the distance. More formally, if d is the smallest non-negative integer so that $\mathcal{C}_1[0, \ell_1 - d] \preceq \mathcal{C}$, then we say the distance between the reference chain $\mathcal{C}$ and the branch $\mathcal{C}_1$ is d, and we write* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C}) = d$.

Remark 2. *Note that the distance of chain $\mathcal{C}$ from chain $\mathcal{C}_1$ is different from the distance of $\mathcal{C}_1$ from $\mathcal{C}$, and it is possible that* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}_1) \neq \mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C})$*. For example, in Figure 20, the distance of $\mathcal{C}$ from $\mathcal{C}_1$ is 4, i.e.* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}_1) = 4$*, while the distance of $\mathcal{C}_1$ from $\mathcal{C}$ is 2, i.e.,* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C}) = 2$*. We also note that the distance of $\mathcal{C}$ from itself is always 0, i.e.,* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}) = 0$.

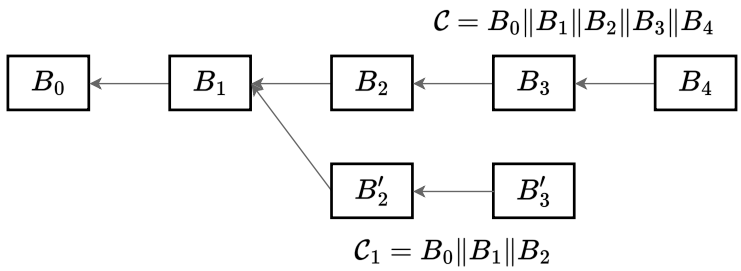


Figure 20: A toy example for illustrating the distance between two chains $\mathcal{C} = B_0 \| B_1 \| B_2 \| B_3 \| B_4$ and $\mathcal{C}_1 = B_0 \| B_1 \| B'_2 \| B'_3$. Here, $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C}) = 2$, i.e., the distance from $\mathcal{C}_1$ to $\mathcal{C}$ is 2. Similarly, $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}_1) = 3$, i.e., the distance from $\mathcal{C}$ to $\mathcal{C}_1$ is 3.

After explaining the concept of the *distance between two chains*, we are ready to introduce the distance-greedy strategies. Intuitively, a player who plays a distance-greedy strategy will make attempts to extend a *set of best chains* in which those chain

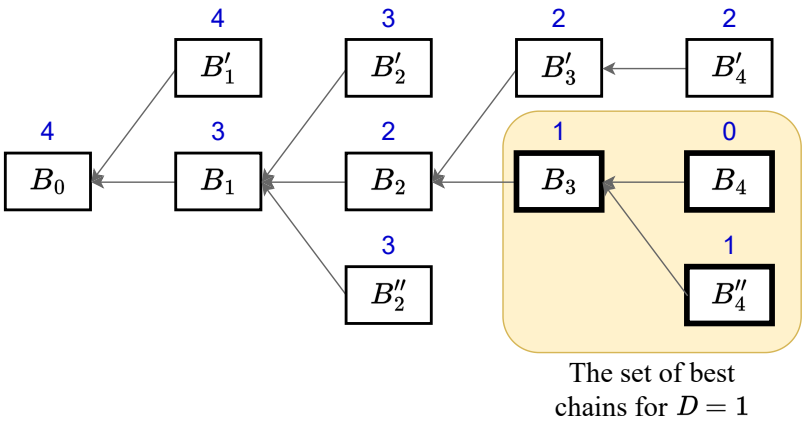


Figure 21: A toy example of 1-distance-greedy strategy. Here, the best chain is $\mathcal{C}_{\text{best}} = B_0\|B_1\|B_2\|B_3\|B_4$. The number in blue on top of each block denotes the distance from the *best chain* $\mathcal{C}_{\text{best}}$ (i.e., the branch chain) to the *reference chain* that consists of a sequence of blocks from the genesis block B_0 to that block. The bold blocks in the yellow area are the last blocks of the chains in the set of best chains.

should be very "close" to the best chain, i.e., the distance from the best chain to the chain must be small. Here, we consider the best chain as the branch chain and all other chains in the set of best chains as the reference chains. By the definition of the distance, we can obtain a common prefix of all reference chains by removing the last few blocks of the branch chain. Jumping ahead, we will use this observation to prove the common prefix property of our protocol. More formally, we have the following definition.

Definition 62 (D-distance-greedy strategy)**.** *Consider a blockchain protocol execution. Let P be a player of the protocol execution, and let $\mathbb{C}$ be the set of chains in player P's local view. Let $\mathcal{C}_{\text{best}}$ be the longest chain at round r, where $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$. Let D be non-negative integers. Define a set of chains $\mathbb{C}_{\text{best}}$ as*

$$\mathbb{C}_{\text{best}} = \left\{\mathcal{C} \in \mathbb{C} \;\middle|\; \mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq D\right\}.$$

We say the player is D-distance-greedy *if, for all* r*, player* P *makes attempts to extend all chains* $\mathcal{C} \in \mathbb{C}_{\text{best}}$.

In Figure 21, a pictorial illustration for the toy example of 1-distance-greedy strategy can be found. The honest players will extend the bold blocks (B_3, B_4, B_4'').

4.3.3 The protocol $\Pi^{\bullet}$

We present a new protocol $\Pi^{\bullet}$ with the goal to achieve the best possible unpredictability while only requiring a much smaller fraction (e.g., 57%) of honest stake to achieve the security properties. To simplify the presentation, we construct a protocol in which the payloads in all blocks are empty. We will extend our protocol to include the payload in Section 4.9.1. Protocol $\Pi^{\bullet}$ uses a unique digital signature scheme and a hash function as building blocks. For completeness, we present the definition of the unique digital signature scheme in Supplemental material 4.7.2.

Blockchain initialization phase. In this phase, the genesis block will be created. Here, the genesis block consists of a randomness, the public keys of the players. Given a group of PoS-players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, a security parameter κ, and a unique digital signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, the initialization is as follows: each PoS-player $P_j \in \mathcal{P}$ generates a key pair $(\text{SK}_j, \text{PK}_j) \leftarrow \mathsf{uKeyGen}(1^{\kappa})$, publishes the public key PK_j and keeps SK_j secret. The public keys are stored in the genesis block of the blockchain system; let B_0 denote the genesis block. In addition, an independent randomness $\mathsf{rand} \in \{0,1\}^{\kappa}$ will also be stored in B_0. That is $B_0 = \langle(\text{PK}_1, \text{PK}_2, \cdots, \text{PK}_n), \mathsf{rand}\rangle$[3].

Blockchain extension phase. Follow the design of multi-extension framework,

[3]For simplicity, we omit the stake distribution in the genesis block since all players have the same number of stake in the flat model.

Algorithm 7: Algorithms $\mathsf{Context}^\bullet$, $\mathsf{Mining}^\bullet$, $\mathsf{Validate}^\bullet$, and D-$\mathsf{BestChainSet}^\bullet$.

$\mathsf{Context}^\bullet(\mathcal{C})$:
  $\ell := \mathsf{len}(\mathcal{C})$; $\eta := h(\mathcal{C}[\ell])$; Return η;
$\mathsf{Mining}^\bullet(\eta, r, \mathrm{SK}, \mathrm{PK})$:
  $\sigma := \mathsf{uSign}(\mathrm{SK}, \langle \eta, r \rangle)$
  if $\mathsf{H}(\eta, r, \mathrm{PK}, \sigma) < \mathtt{T}$ **then**
    Create new block $B := \langle \eta, r, \mathrm{PK}, \sigma \rangle$; Return B;
  else Return $\perp$
$\mathsf{Validate}^\bullet(\mathcal{C}, r)$:
  Parse $\mathcal{C}$ into $B_0 \| B_1 \| \cdots \| B_\ell$;
  for $i \in [1, \ell]$ **do**
    Parse B_i into $\langle \eta_i, r_i, \mathrm{PK}_i, \sigma_i \rangle$;
    if $h(B_{i-1}) \neq \eta_i$ *or* $\mathsf{H}(\eta_i, r_i, \mathrm{PK}_i, \sigma_i) \geq \mathtt{T}$ *or* $\mathsf{uVerify}(\mathrm{PK}_i, \langle \eta_i, r_i \rangle, \sigma_i) = 0$ *or* $r_i > r$ **then**
      Return 0;
  Return 1;
D-$\mathsf{BestChainSet}^\bullet(\mathbb{C})$:
  Set $\mathcal{C}_{\mathrm{best}}$ as the longest chain in $\mathbb{C}$ and $\mathbb{C}_{\mathrm{best}} = \{\mathcal{C}_{\mathrm{best}}\}$;
  for $\mathcal{C} \in \mathbb{C}$ **do**
    if $\mathsf{distance}(\mathcal{C}_{\mathrm{best}} \to \mathcal{C}) \leq D$ **then**
      $\mathbb{C}_{\mathrm{best}} := \mathbb{C}_{\mathrm{best}} \cup \{\mathcal{C}\}$;

our protocol is parameterized by four algorithms: $\mathsf{Context}^\bullet$, $\mathsf{Mining}^\bullet$, $\mathsf{Validate}^\bullet$, and D-$\mathsf{BestChainSet}^\bullet$. (Please see Algorithm 7 for the pseudocode of the four algorithms.) In our protocol, the players extend a set of chains $\mathbb{C}_{\mathrm{best}}$ in which, for all chain $\mathcal{C} \in \mathbb{C}_{\mathrm{best}}$, the distance from the best chain $\mathcal{C}_{\mathrm{best}}$ to the chain $\mathcal{C}$ does not exceed D, i.e., $\mathsf{distance}(\mathcal{C}_{\mathrm{best}} \to \mathcal{C}) \leq D$.

The procedure D-$\mathsf{BestChainSet}^\bullet$ will output *a set of best chains* including the longest (i.e., the best) chain, and several chains that are very close to the longest chain. First, the procedure D-$\mathsf{BestChainSet}^\bullet$ iterates through all chains in the local state and uses algorithm $\mathsf{Validate}^\bullet$ to remove the invalid chains. Here, the algorithm $\mathsf{Validate}^\bullet$ takes a chain $\mathcal{C}$ and the current round r as input and evaluates every block of the chain $\mathcal{C}$ sequentially. Let ℓ be the length of $\mathcal{C}$. Starting from the head of $\mathcal{C}$, for every block $\mathcal{C}[i]$, for all $i \in [\ell]$, in the chain $\mathcal{C}$, the procedure D-$\mathsf{BestChainSet}^\bullet$ verifies

that 1) $\mathcal{C}[i]$ is linked to the previous block $\mathcal{C}[i-1]$ correctly, 2) the hash inequality is correct, and 3) the signature is correctly generated by the player. Then, the procedure D-$\mathsf{BestChainSet}^\bullet$ selects the best chain $\mathcal{C}_{\text{best}}$ as the longest chain and iterates through the set of chains in the local state of the player to find all the chains in which the distances from the best chain to those chains do not exceed D.

For a chain $\mathcal{C} = B_0 \| B_1 \| B_2 \| \ldots \| B_i$ in the set of best chains $\mathbb{C}_{\text{best}}$, the honest players P, with key pair (SK, PK), make attempts to extend the chain $\mathcal{C}$ as follows. Let r denote the current time (or round number). The player P first computes the context $\eta := \mathsf{Context}^\bullet(\mathcal{C})$. Here, algorithm $\mathsf{Context}^\bullet$ return the hash value of the last block on $\mathcal{C}$, i.e., $\mathsf{Context}^\bullet(\mathcal{C}) = h(B_i)$. A PoS-player P is able to successfully generate a new block if the following hash inequality holds: $\mathsf{H}(\eta, r, \text{PK}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}(\text{SK}, \langle \eta, r \rangle)$. The new block B_{i+1} is defined as $B_{i+1} := \langle \eta, r, \text{PK}, \sigma \rangle$.

Algorithm 8: PROTOCOL $\Pi^\bullet$

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with key pair (SK, PK) and local set of chains $\mathbb{C}$, proceeds as follows.

1 Upon receiving a chain $\mathcal{C}'$, set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$ after verifying $\mathsf{Validate}^\bullet(\mathcal{C}', r) = 1$;
2 Compute $\mathbb{C}_{\text{best}} := D$-$\mathsf{BestChainSet}^\bullet(\mathbb{C})$;
3 **for** $\mathcal{C} \in \mathbb{C}_{\text{best}}$ **do**
4 $\quad \eta := \mathsf{Context}^\bullet(\mathcal{C})$; $B := \mathsf{Mining}^\bullet(\eta, r, \text{SK}, \text{PK})$;
5 $\quad$ **if** $B \neq \perp$ **then**
6 $\quad\quad \mathcal{C}_1 := \mathcal{C} \| B$; Broadcast $\mathcal{C}_1$;

4.3.4 A new tiebreak rule for our multi-extension protocol

We design a new tiebreak rule for our D-greedy strategy. In a multi-extension protocol, the honest players extend all chains in the set (of best chains). The probability of generating a new best chain can vary depending on the number of chains in the set of best chains. This opens up opportunities for an adversary to slow down the chain growth by publishing a chain with the same length as the best chain but

with fewer chains in the set of best chains. To defend against this attack, it is crucial to establish a tiebreak rule that maximizes the growth of the chain. In contrast, the probability of generating a new best chain in a single-extension protocol is constant; thus such tiebreak rule (that maximizes the growth) is not needed in single-extension PoS protocols.

Intuitively, when there are two equally longest chains in a round, the best chain is selected based on the expected time to extend the chain and generate a new best chain. Honest players will choose the chain that is expected to be extended more quickly.

Recall that, the probability of generating a new best chain can vary depending on the number of chains in the set. As the number of chains in the set of best chains increases, the chance for honest nodes to generate a new longest chain also increases. To take advantage of this, our tiebreak rule prioritizes the chain with more chains in the set of best chains. This guarantees that the adversary cannot slow down the chain growth of the honest players.

Depth-based subsets. Before presenting the tiebreak rule, we introduce the definition of the *depth-based subsets*. Consider a protocol execution at a certain round, let $\mathcal{C}_{\text{best}}$ denote the best chain and $\mathbb{C}_{\text{best}}$ be the set of best chains. In our protocol execution, honest players follow the D-greedy strategy, and make attempts to extend the set of best chains. As shown in Figure 22, we partition the set $\mathbb{C}_{\text{best}}$ into $D+1$ number of *disjoint subsets* based on the length of those chains. Let $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$ be the length of the best chain, and for all $i \in [0..D]$, the i-depth-based subset L_i is the subset of chains with the length of $\ell - i$ in the set $\mathbb{C}_{\text{best}}$. That is, $L_i = \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$.

Our new tiebreak rule. The tiebreak rule states that when there are two chains of

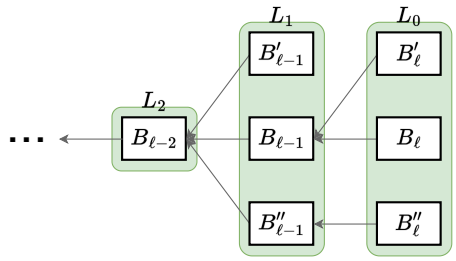


Figure 22: Partitioning a set of best chains $\mathbb{C}_{\text{best}}$ into multiple disjoint depth-based subsets for $D = 2$. Here, the set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into 3 subsets L_0, L_1, L_2. Let ℓ be the length of the best chain. The 0-depth subset L_0 consists of 3 chains of length ℓ, i.e., the chains that have the last blocks are $B_\ell, B'_\ell, B''_\ell$. The 1-depth subset L_1 consists of 3 chains of length $\ell - 1$, i.e., the chains that have the last blocks are $B_{\ell-1}, B'_{\ell-1}B''_{\ell-1}$. The 2-depth subset L_2 consists of 1 chain of length $\ell - 2$, i.e., the chain that has the last block is $B_{\ell-2}$.

the same length, the one with a faster expected time for further extension is chosen. Recall that, in our protocol, honest players extend all chains in the set of best chain. The tiebreak rule is based on the number of chains in the set. By utilizing this rule, we can ensure that the adversary is unable to slow down the growth of the chain for honest players.

In our protocol, honest players use a D-greedy strategy to extend the set of best chains. The length of the best chain increases by 1 when a chain in the 0-depth subset (i.e., a chain with the same length as the best chain) is extended. Therefore, having more chains in the 0-depth subset will allow honest players to extend the best chain faster. Additionally, when a chain in the 1-depth subset is extended, the number of chains in the 0-depth subset also increases. As a result, if the number of chains in the 0-depth subset is the same, having more chains in the 1-depth subset will also allow honest players to extend the best chain faster.

To break the tie of two equally longest chains, the players compare the number of chains in subsets at 0-depth, 1-depth, 2-depth, and so on, in order to determine the best chain (see Algorithm 9 for the pseudocode). If the number of chains in subsets is the same, we break the tie by comparing the number of chains in the next depth subset, and so on. If the tie still cannot be broken, the chain with the smallest hash value of the last block is chosen as the best chain.

Algorithm 9: Tiebreak rule

Input : Two chains $\mathcal{C}_{\text{best}}$, $\mathcal{C}'_{\text{best}}$ of length ℓ, the local set of chains $\mathbb{C}$
Output: Return the better chain between $\mathcal{C}_{\text{best}}$ and $\mathcal{C}'_{\text{best}}$

$\mathbb{C}_{\text{best}} := \emptyset$; $\mathbb{C}'_{\text{best}} := \emptyset$
for $\mathcal{C} \in \mathbb{C}$ **do**
  if $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq D$ **then**
    $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$
  if $\mathsf{distance}(\mathcal{C}'_{\text{best}} \to \mathcal{C}) \leq D$ **then**
    $\mathbb{C}'_{\text{best}} := \mathbb{C}'_{\text{best}} \cup \{\mathcal{C}\}$
for $i \in [0..D]$ **do**
  $L_i := \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$
  $L'_i := \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$
$i := 0$
while $i \leq D$ **do**
  if $|L_i| > |L'_i|$ **then**
    Return $\mathcal{C}_{\text{best}}$
  if $|L_i| < |L'_i|$ **then**
    Return $\mathcal{C}'_{\text{best}}$
  if $|L_i| = |L'_i|$ **then**
    $i := i + 1$
if $\mathsf{H}(\mathcal{C}_{\text{best}}[\ell]) < \mathsf{H}(\mathcal{C}'_{\text{best}}[\ell])$ **then**
  Return $\mathcal{C}_{\text{best}}$
else
  Return $\mathcal{C}'_{\text{best}}$

More concretely, suppose we have two equally longest chains $\mathcal{C}$ and $\mathcal{C}'$. Let $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'_{\text{best}}$ be the corresponding sets of best chains for $\mathcal{C}$ and $\mathcal{C}'$, respectively. For $i \in [0..D]$, let L_i denote the set of chains in $\mathbb{C}_{\text{best}}$ with length $\ell - i$, and let L'_i denote the set of chains in $\mathbb{C}'_{\text{best}}$ with length $\ell - i$. In other words, L_i and L'_i are i-depth subsets in $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'_{\text{best}}$, respectively. To break the tie between $\mathcal{C}$ and $\mathcal{C}'$,

the players follow this procedure: If the number of chains in L_0 is bigger than the number of chains in L'_0, i.e., $|L_0| > |L'_0|$, we say the chain $\mathcal{C}$ is better than the chain $\mathcal{C}'$. Similarly, if $|L'_0| > |L_0|$, we say the chain $\mathcal{C}'$ is better than the chain $\mathcal{C}$. If $|L_0| = |L'_0|$, we compare the number of chains in L_1, L'_1. If $|L_1| > |L'_1|$, we say we say the chain $\mathcal{C}$ is better than the chain $\mathcal{C}'$. Similarly, if $|L'_1| > |L_1|$, we say the chain $\mathcal{C}'$ is better than the chain $\mathcal{C}$. If $|L_1| = |L'_1|$, we compare the number of chains in L_2, L'_2, and so on. If we still cannot break the tie, we choose the chain with the smallest hash value of the last block.

4.3.5 Addressing the tradeoff on security and performance

When the greedy parameter D increases, the number of chains in the set of best chains also increases. This means that honest players are more likely to create the longest chain, which ultimately reduces the amount of total stake controlled by honest players needed to maintain security. However, this increase in the number of chains also has drawbacks on performance. Specifically, players must download additional blocks that are not part of the best chain. As the transactions in those blocks are not included in the ledgers, the resources (communication and storage) used for those blocks are wasted. In other words, if the greedy parameter D increases, players need to spend more resources to maintain the blockchain, but the overall performance (throughput) does not improve.

To address this issue, a directed acyclic graph (DAG) can be used to link blocks on the best chain to additional blocks. DAG technology allows transactions within these blocks to be included in the ledger, ensuring that the resources required to download them are not wasted. DAG design provides a way to maintain network security without sacrificing efficiency. Furthermore, by enabling parallel transactions within multiple blocks, DAG technology can increase the transaction rate of the

network. This can be especially beneficial for high-volume networks where efficiency is a primary concern..

4.4 Security Analysis: Overview

In this section, we provide the overview of security analysis for protocol $\Pi^{\bullet}$. Then, in Section 4.5, we propose a new analysis framework to study the chain growth in multi-extension protocols. In Section 4.6, we use the above analysis framework to examine the chain growth of our protocol. In Section 4.7, we present a new analysis framework to analyze the common prefix property. Finally, in Section 4.8, we present the analysis of chain quality and the best possible unpredictability.

As in the previous section, assuming the underlying scheme ($\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify}$) is a unique digital signature scheme, a malicious player for a given context, can create *exactly one* signature. Now, we can prove the security properties of protocol $\Pi^{\bullet}$ under the assumption of honest majority of *effective stake* based on $\alpha^{\bullet} = \hat{\mathtt{A}}^{\bullet}_{D} \cdot \alpha$ (e.g., $\hat{\mathtt{A}}^{\bullet}_{50} \geq 2.04$) and $\beta^{\bullet} = 2.72\beta$.

It is important to note that the techniques described in [47, 88, 34, 7] can provide valuable insights for analyzing the security properties of protocols based on the single extension design framework. However, our protocol $\Pi^{\bullet}$ *does not* follow this framework and requires new analysis techniques to prove its security properties.

Theorem 63. *Consider an execution of multi-extension protocol $\Pi^{\bullet}$ in the random oracle model, where honest players follow the D-distance-greedy strategy while adversarial players could follow any arbitrary strategy. Additionally, all players have their stake registered at the beginning of the execution. Assume* ($\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify}$) *is a unique digital signature scheme, and $\alpha^{\bullet} = \lambda\beta^{\bullet}$, $\lambda > 1$. Then protocol $\Pi^{\bullet}$ achieves 1) chain growth, chain quality, and common prefix properties; and 2) the best possible unpredictability.*

Chain growth. We propose a new analysis framework to study the chain growth in multi-extension protocols in Section 4.5. We develop *a random walk in a Markov chain that consists of multiple states* to analyze the chain growth property. In the Markov chain, each state provides a representation of the set of best chains in a protocol round. Note that, for the existing single-extension protocols [47, 88, 34], since the honest players only extend a single best chain, the probability that honest players extend the best chain is the same for every round. Hence, the analysis of chain growth for such protocols is quite simple. On the other hand, in multi-extension protocols, the honest players may extend multiple chains in a single round, and the probability of extending the best chain can vary between rounds. Therefore, a new analysis framework is necessary to evaluate the chain growth in multi-extension protocols.

In Section 4.6, we use the above analysis framework to examine the chain growth of our protocol. Note that, in a multi-extension protocol, the adversary may attempt to slow down the chain growth by launching attacks. Fortunately, as we mention in the previous paragraph, our tiebreak rule prevents the adversary from launching such attacks. We start our analysis with the design of a *simplified Markov chain* and then extend it to design an *augmented Markov chain*.

Simplified Markov chain. We design the simplified Markov chain using the information of the depth-based subsets. Recall that, by following the D-distance-greedy strategy, the honest players extend a set of best chains. The set of best chains can be partitioned into $D + 1$ subsets based on the depth of those chains, where the depth of a chain is computed based on the difference between its length and the length of the current best chain. For $i \in [0..D]$, the i-depth subset consists of all the chains that are i blocks behind the best chain. In each round, a new best chain is generated if a player extends a chain in the 0-depth subset, which consists of all the chains that have the same length as the current best chain. Further, a new chain is added to the i-depth

subset if a chain in the $(i+1)$-depth subset is extended, where $i \in [0..D-1]$. Hence, we can analyze the chain growth based on the number of chains in those subsets. In the simplified Markov chain, each state represents a protocol round with specific numbers of chains in all depth-based subsets. The chain growth of our protocol can be estimated based on the expected number of chains in the 0-depth subset. The simplified Markov chain provides information about the number of chains in depth-based subsets, but it does not provide how many chains are removed from the set of best chains when a new best chain is generated. This leads to a worst-case scenario where the set of best chains only consists of the best chain and its prefixes, making it difficult to determine a good lower bound for the amplification ratio, even with a large D.

Augmented Markov chain. To resolve the issue in the simplified Markov chain, we introduce an augmented Markov chain. The states of the augmented Markov chain contain more information. This helps us determine the number of chains that are removed after generating a new best chain. By doing so, we can avoid considering the worst-case scenario where the set of best chains only consists of the best chain and its prefixes. More concretely, we present the notion of *depth-distance-based subsets*. These subsets are selected based on *both* the length of the chains and their distance from the best chain. When a new best chain is generated, the distance from the new best chain to the chains in the subsets increases by one. As a result, we can obtain the number of chains in the new depth-distance-based subsets (when the new best chain is generated) based on the number of chains in the old depth-distance-based subsets (when the new best chain has not been generated). The states in the augmented Markov chain provide information about the number of chains in the depth-distance-based subset, allowing us to identify which chains belong to the new set of best chains when a new best chain is generated. This approach results in a better lower bound

on the amplification ratio.

Common prefix. To analyze the common prefix property, we first demonstrate that the adversary can increase their stake by a factor of $e = 2.72$ if they extend the chain themselves. It is worth noting that the adversary cannot use the blocks of honest players to compromise the security property. To break the common prefix property, the adversary must find a way to create two divergent chains, i.e., two chains that do not share a common prefix after removing the last $\kappa + D$ blocks. Recall that, in our protocol, we use the D-distance-greedy strategy, where honest players in the protocol execution will only extend the set of best chains. Based on the definition of the D-distance-greedy strategy, the set of best chains must be "close". That is, these chains share a common prefix after removing the last D blocks. Thus, the chains produced by the honest players share a common prefix with the best chain, after removing the last D blocks. This prevents the adversary from using the chain of honest players to break the common prefix property.

To formally prove the common prefix property, we introduce the notions of *virtual block-sets* and *virtual chains*, and then define the *common prefix property w.r.t. virtual chains*. We can prove the common prefix w.r.t. virtual chains by showing that the honest players only contribute at most one virtual block-set at a block height. Afterward, we show that the standard common prefix property can be reduced to common prefix w.r.t. virtual chains.

Virtual block-sets and virtual chains. A virtual block-set consists of multiple blocks with the same height that are "close" to each other. More concretely, we first define two chains as "close" if they share a common prefix after removing the last few blocks, say D, where D is a parameter as mentioned above. When two chains are "close", the last blocks of the two chains are also "close". Now the virtual chain consists of

multiple virtual block-sets that are linked together.

Common prefix w.r.t. virtual chains. To prove the common prefix w.r.t. virtual chains, we introduce the concept of *honest virtual block-sets*. We say a virtual block-set is honest if the first generated block in the virtual block-set is generated by an honest player. As the honest players follow a D-greedy strategy that we mentioned above, at each block height, there is at most one honest virtual block-set. Thus, to break the common prefix property w.r.t virtual chains, the adversary needs to generate more virtual block-sets than the honest players. This requires the adversary to control the majority of the stake. Thus contradicts the assumption that honest players control the majority of the stake.

Common prefix. Finally, we demonstrate that the common prefix property can be achieved from the common prefix w.r.t. virtual chains. Recall that, the common prefix w.r.t. virtual chains property states that the best virtual chains of honest players share the same common prefix after removing the last κ virtual block-sets. Plus, based on the definition of the virtual block-set, the blocks in the same virtual block-set are "close" together. In other words, the chains that have those block as the last blocks share the common prefix after removing the last D blocks. Hence, the best chains of honest players share the same common prefix after removing the last $\kappa + D$ blocks.

Chain quality. After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [88]. Intuitively, in order to break the chain quality property, the adversary must generate κ consecutive blocks on the best chains. This requires the adversary to control majority of the stake.

The best possible unpredictability. In our protocol, the players extract the context of a chain based on the hash value of the last block. Thus, a player cannot

know the hash value of the next block unless he generate the block himself. Hence, the player can only predict whether or not she/he can generate the next block. In other words, our protocol achieves the best possible unpredictability.

4.5 Chain Growth in Multi-Extension: A New Analysis Framework

In this section, we present a framework for analyzing the chain growth property in multi-extension protocols using the Markov chain. We develop a random walk in a Markov chain. The Markov chain consists of multiple states, where each state provides some information on the set of best chains in a protocol round. Then, we will apply this framework to analyze the chain growth property of our protocol $\Pi^{\bullet}$.

We construct a *Markov chain* with a *state space* S and a *transition matrix* $\mathbf{T}$. Each state $s \in S$ provides some information on the view of the players in a protocol round. The transition matrix $\mathbf{T}$ is a $|S| \times |S|$ matrix that reflects how the set of best chains is updated after one protocol round. We define a *chain growth function* $\mathsf{growth} : S \to [0, 1]$ to represent the chain growth rate on each state. Then, we consider a *random walk* of t states $s_1, s_2, \cdots, s_t$, where $t \in \mathbb{N}$ and for all $i \in [t], s_i \in S$. This random walk represents how the set of best chains is updated in t protocol rounds. The chain growth is computed as the sum of outputs of the chain growth function over all the states in the random walk. (See Section 4.5.1 for more details.)

We remark that, in a single-extension protocol, the probability of honest players extending the best chain remains constant in every round. As a result, the chain growth function can be simplified to a constant value, making the analysis of chain growth property easier, compared to the analysis of a multi-extension protocol.

4.5.1 Defining a Markov chain

Before presenting our analysis for the chain growth property, let us summarize the definition of a Markov chain and the random walk on a Markov chain [27]. We will use the Markov chain to analyze the chain growth property of our protocol.

Markov chain. A *Markov chain* is a mathematical model that describes a sequence of events in which the probability of each event depends only on the state preceding it. The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed. In other words, the probability of transitioning to any particular state is dependent solely on the current state. A Markov chain is specified by a *state space* S and a *transition matrix* $\mathbf{T}$. For simplicity, we will refer to the Markov chain as $(S, \mathbf{T})$. Each state $s \in S$ is a tuple of d integers $\langle n_0, \cdots, n_{d-1} \rangle$, where $d \in \mathbb{N}$ and for all $i \in [0..d], n_i \in \mathbb{N}$. In our analysis, each state provides some information on the view of the players in a protocol round. For example, we can define each state $s \in S$ in the format of $\langle n_0 \rangle$, where $n_0 \in \mathbb{N}$. Here, n_0 is the number of chains that have the same length as the current best chain. The state space is given as $S = \{\langle n_0 \rangle\}_{n_0 \in \mathbb{N}}$. The transition matrix $\mathbf{T}$ is an $|S| \times |S|$ matrix that contains information on the probability of transitioning between states, where $|S|$ is the size of the state space S. For any two states s and s', the probability of transitioning from state s to state s' is $\mathbf{T}_{s,s'}$.

Random walk. A *random walk* in the Markov chain is a sequence of t states $s_1, s_2, \cdots, s_t$, where $t \in \mathbb{N}$ and for all $i \in [t]$, $s_i \in S$. The random walk starts at some state s_1, traverses to a new state s_2, based on the transition matrix $\mathbf{T}$, and then repeats the process.

Stationary distribution. A *stationary distribution* $Q = [q_s]_{s \in S}$ of a Markov chain is a probability distribution that represents the probabilities that states appear in

a random walk. Here, the probability that a state s is drawn from the stationary distribution Q is q_s. If the state s is randomly drawn from the stationary distribution Q, we write $s \sim Q$. The sum of the probabilities in Q equals 1, i.e., $\sum_{s \in S} q_s = 1$. For every state $s \in S$, the probability q_s is computed based on the transitions from other states to the state s, i.e., $q_s = \sum_{s' \in S} q_{s'} \cdot \mathbf{T}_{s',s}$. Hence, we can obtain the stationary distribution by solving the following equations.

$$\begin{cases} \sum_{s \in S} q_s = 1, \\ q_s = \sum_{s' \in S} \left(q_{s'} \cdot \mathbf{T}_{s',s} \right), \forall s \in S. \end{cases} \tag{4.1}$$

Chain growth function. We define a *chain growth function* $\mathsf{growth} : S \to [0, 1]$ to represent the chain growth rate on each state. For example, if each state $s \in S$ is in the format of $\langle n \rangle$, where n is the number of chains that have the same length with the current best chains, we have, $\mathsf{growth}(n) = n \cdot \alpha$. Consider a random walk $s_1, s_2, \cdots, s_t$. The chain growth, i.e., the increasing length of the best chain, in those t protocol rounds is computed as $\sum_{i=1}^{t} \mathsf{growth}(s_i)$. As the stationary distribution Q represents the probabilities that states appearing in a random walk, the expected chain growth in a protocol is given by

$$\bar{g} = \mathbb{E}_{s \sim Q}[\mathsf{growth}(s)].$$

Compatible Markov chain and chain growth function for a protocol execution. To analyze the chain growth of a multi-extension protocol, we need to design a *compatible* Markov chain and chain growth function. We say the Markov chain $(S, \mathbf{T})$ and the chain growth function growth is *compatible* to the execution of protocol Π° if for every state $s \in S$ that represents the view of the players at round r, the probability that the honest players generate a new best chain is at least $\mathsf{growth}(s)$.

Definition 64 (A compatible Markov chain and chain growth function for a protocol execution)**.** *Consider a multi-extension proof-of-stake protocol* Π°*. Consider a Markov chain* $(S, \mathbf{T})$ *and a chain growth function* $\mathsf{growth} : S \to [0, 1]$*. For a protocol round* r*, the view* $\mathtt{VIEW}^r$ *of players at round* r *can be mapped to a state* s *in the state space* S*. We say the Markov chain* $(S, \mathbf{T})$ *and the chain growth function* $\mathsf{growth} : S \to [0, 1]$ *are typical to the execution of protocol* Π° *if for every round* r *with the view* $\mathtt{VIEW}^r$*, which is represented by a state* $s \in S$*, we have the probability that the length of the best chain increases by* 1 *at round* r *is at least* $\mathsf{growth}(s)$*.*

4.5.2 Chain growth property for a multi-extension protocol

Consider a multi-extension proof-of-stake protocol Π°. Consider a Markov chain $(S, \mathbf{T})$ and a chain growth function $\mathsf{growth} : S \to [0, 1]$ that are compatible to the execution of the protocol Π°. We can bound the chain growth, i.e., the increasing length of the best chain, in a multi-extension proof-of-stake protocol Π° using the compatible Markov chain and chain growth function as follows.

Lemma 65 (Chain growth property for a multi-extension protocol)**.** *Consider a Markov chain* $(S, \mathbf{T})$ *and a chain growth function* $\mathsf{growth} : S \to [0, 1]$ *that are compatible to a multi-extension protocol* Π°*. Let* Q *be the stationary distribution over* S*, and* $\bar{g} = \mathbb{E}_{s \sim Q}[\mathsf{growth}(s)]$ *be the expected chain growth. Consider an honest player* P *with the best chain* $\mathcal{C}$ *in round* r*, and an honest player* P_1 *with the best chain* $\mathcal{C}_1$ *in round* r_1*, where* $r_1 = r + t$*, for some* $t = \Omega(\kappa)$*. Then we have* $\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq (1 - \delta) \cdot \bar{g} \cdot t\right] \geq 1 - e^{-\Omega(\kappa)}$ *where* $t = r_1 - r$*, and* $\delta > 0$*.*

Proof. Consider a random walk from round r to round r_1. Let s_i denote that state at round i in the random walk, where $i \in [r..r_1]$ and $s_i \in S$. Since the Markov chain $(S, \mathbf{T})$ and the chain growth function $\mathsf{growth} : S \to [0, 1]$ that are compatible

to protocol Π°, the probability that the players extend the best chain at round i is at least $\mathsf{growth}(s_i)$. Using the Chernoff bound on the Markov chain in [27], we have,

$$\Pr[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) < (1-\delta) \cdot \bar{g} \cdot t] < e^{-\Omega(\kappa)}.$$

□

We remark that, our analysis framework can be applied to other multi-extension protocols, beyond the protocol in Section 4.3.3. By designing the compatible Markov chain, and chain growth function, we can use the Chernoff bound to analyze the chain growth over a sufficiently long period. This approach provides a general and flexible way to study the chain growth of multi-extension protocols.

4.6 Chain Growth in Multi-Extension: Security analysis details

We will now analyze the chain growth property of our protocol using the analysis framework in Section 4.5. Before constructing the Markov chains for our protocol, we will first consider a hybrid execution in which the malicious players will not contribute to the chain extension. We demonstrate that by utilizing the tiebreak method, honest players will always follow the best chain with the fastest expected time for extension. As a result, the adversary is unable to slow down the growth of the chain for honest players. The chain growth in this hybrid scenario serves as a lower bound for the chain growth in the real execution. We will then construct two Markov chains to analyze the chain growth in the hybrid execution for the protocol.

We start with a simplified Markov chain. Then, we extend the simplified Markov chain to design an augmented Markov chain. In Figure 23, we show the lower bounds of the amplification ratio using the simplified and augmented Markov chains. In Figure 24, we show the corresponding fraction of honest stake to prove the security

of the protocol based on the lower bounds of the amplification ratio. For example, by using the augmented Markov chain, we have $\hat{\mathtt{A}}^{\bullet}_{50} \geq 2.04$, i.e., protocol $\Pi^{\bullet}$ is secure if 57% of stake is honest.

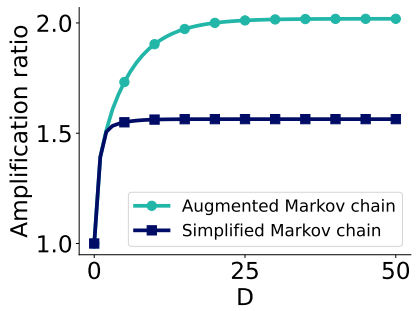


Figure 23: The lower bounds of the amplification ratio using the simplified and augmented Markov chains resepectively.

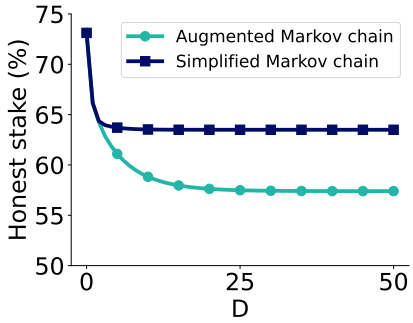


Figure 24: The upper bounds of the fraction on honest players using the simplified and augmented Markov chains resepectively.

We facilitate the analysis of the chain growth property for our protocol by defining a new notion, called *amplification ratio*. The amplification ratio is the ratio between the chain growth when the honest players follow D-distance-greedy strategy and 0-distance-greedy strategy. In our protocol, in each round, for each chain in the set of best chains, each honest player makes one attempt to generate a new block by making one query to the random oracle. The event where an honest player successfully generates a new block (from a given chain in the set of best chains) can be modeled as an (independent) Bernoulli random variable which takes the value 1 with probability $p = \frac{\mathtt{T}}{2^{\kappa}}$. Hence, in each round, the probability that an honest player extends a chain in the set of best chains is $\alpha = 1 - (1-p)^{N\cdot(1-\rho)}$. Let N_0 and N_D be the average increased length of the longest chain that is extended by the honest players, following the 0-distance-greedy, and D-distance-greedy strategies, respectively. In

the 0-distance-greedy, since the honest players only extend the best chain, we have, $N_0 = \alpha$. We define the amplification ratio in the presence of an adversary for the D-distance-greedy strategy as $\mathtt{A}^{\bullet}_{D} = \frac{N_D}{\alpha}$. We can compute the amplification ratio is as

$$\hat{\mathtt{A}}^{\bullet}_{D} = \frac{\bar{g}}{\alpha} = \mathbb{E}_{s\sim Q}\left[\frac{\mathsf{growth}(s)}{\alpha}\right].$$

4.6.1 A hybrid experiment: Ignoring the adversarial extension

We consider a hybrid experiment where all messages sent by the adversary are removed. Through this experiment, we demonstrate that the adversary cannot slow down the growth of the honest player's chain. We note that, hybrid experiments were introduced in the analysis of the Bitcoin protocol in [88].

Let $\mathsf{REAL}(\omega) = \mathsf{EXEC}_{\Pi^{\bullet},\mathcal{A},\mathcal{Z}}(\omega)$ denote the standard execution of $\Pi^{\bullet}$, where ω is the randomness involved in the execution. Let $\mathsf{HYB}^{r}(\omega)$ denote the hybrid execution, which is identical to the real execution up until round r, with the following modifications: 1) the randomness is fixed to ω, and 2) honest players eliminate all new messages sent by the adversary.

No slow-down tiebreak. First, we show that by using the tiebreak rule in Subsection 4.3.4, the adversary cannot slow down the chain growth of honest players. More specifically, we will demonstrate that, based on the tiebreak rule, honest players always choose the best chain that can be extended more quickly.

We consider the expected time for the honest players to generate a new best chain by extending a set of best chain $\mathbb{C}_{\text{best}}$. The set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L_0, L_1, \cdots, L_D$. The length of the best chain increases by 1 if a chain in the subset L_0 (i.e., a chain that has the same length as the best chain) is extended. Plus, for $i \in [0..D-1]$, the number of chains in a subset L_i increases by 1 if a chain in the subset L_{i+1} is extended.

Let $\mathsf{w}(n) = 1-(1-p)^{N\cdot n\cdot(1-\rho)}$ be the probability that the honest players generate at least one block by extending at n number of blocks in a round, where N is the number of players and ρ is the fraction of malicious players. Here, $\mathsf{w}(n) \approx n \cdot \alpha$. For $i \in [0..D]$, let $n_i = |L_i|$ be the number of chains in i-depth subset. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0) \approx n_0 \cdot \alpha$. For $i \in [0..D-1]$, the probability that the honest players generate a new chain in L_i is $\mathsf{w}(n_{i+1}) \approx n_{i+1} \cdot \alpha$.

Let $\mathsf{BlockTime}(n_0, \cdots, n_D)$ be the expected time for the honest players to generate a new best chain by extending a set of best chains $\mathbb{C}_{\text{best}}$. We consider three following cases when the honest players extend the set of best chains $\mathbb{C}_{\text{best}}$.

- *Case 1:* The honest players can generate a new best chain after one round with a probability of $\mathsf{w}(n_0)$.
- *Case 2:* The honest players generate a new chain which is added to the i-depth subset, where $i \in [0..D-1]$. The probability that a chain is added to the i-depth subset is $\mathsf{w}(n_{i+1})$.
- *Case 3:* The honest players do not generate a new chain with a probability of $(1 - \sum_{i=0}^{D-1} \mathsf{w}(n_{i+1}))$.

Based on the three cases above, we can calculate the expected time for the honest players to generate a new best chain as follows.

$$\mathsf{BlockTime}(n_0, \cdots, n_D) = \sum_{i=0}^{D-1} \left(\mathsf{w}(n_{i+1}) \cdot \mathsf{BlockTime}(n_0, \cdots, n_{i-1}, n_i + 1, n_{i+1}, n_D) \right) + (1 - \sum_{i=0}^{D-1} \mathsf{w}(n_{i+1})) \cdot \mathsf{BlockTime}(n_0, \cdots, n_D) + 1.$$

By simplify the above equation, we have,

$$\begin{aligned}
&\left(\sum_{i=0}^{D-1} \mathsf{w}(n_{i+1})\right) \cdot \mathsf{BlockTime}(n_0, \cdots, n_D) \\
=&\sum_{i=0}^{D-1} \left(\mathsf{w}(n_{i+1}) \cdot \mathsf{BlockTime}(n_0, \cdots, n_{i-1}, n_i + 1, n_{i+1}, n_D)\right) + 1 \\
>&\sum_{i=0}^{D-1} \left(\mathsf{w}(n_{i+1}) \cdot \mathsf{BlockTime}(n_0, \cdots, n_{i-1}, n_i + 1, n_{i+1}, n_D)\right).
\end{aligned}$$

Thus, for any $i \in [0..D]$, we have,

$$\mathsf{BlockTime}(n_0, \cdots, n_D) > \mathsf{BlockTime}(n_0, \cdots, n_{i-1}, n_i + 1, n_{i+1}, n_D). \tag{4.2}$$

Base on Equation 4.2, we can show that honest players always choose the best chain that can be extended more quickly.

Lemma 66. *We consider two chains, $\mathcal{C}$ and $\mathcal{C}'$. Let $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'$best be the corresponding sets of best chains for $\mathcal{C}$ and $\mathcal{C}'$, respectively. In other words, the distance from $\mathcal{C}$ to the chains in $\mathbb{C}_{\text{best}}$ is smaller than D, and the distance from $\mathcal{C}'$ to the chains in $\mathbb{C}'_{\text{best}}$ is smaller than D. Here, $\mathbb{C}_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L_0, L_1, \cdots, L_D$ and $\mathbb{C}'_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L'_0, L'_1, \cdots, L'_D$. For $i \in [0..D]$, let $n_i = |L_i|$ be the number of chains in L_i and $n'_i = |L'_i|$ be the number of chains in L'_i. If the tiebreak rule in Algorithm 9 return the chain $\mathcal{C}$, then, we have,*

$$\mathsf{BlockTime}(n_0, \cdots, n_D) < \mathsf{BlockTime}(n'_0, \cdots, n'_D).$$

Analyzing chain growth in the hybrid experiment. Next, we show that the chain growth rate in the execution $\mathsf{REAL}(\omega)$ is always bigger than or equal to the chain growth rate in the execution $\mathsf{HYB}^{\mathcal{F}}(\omega)$. The tiebreak rule ensures that honest players always follow the best chain with the fastest expected time to extend it. Therefore, if

the adversary broadcasts its chain to the honest players, the growth of the best chain will speed up, making it impossible for the adversary to slow down the chain growth of the honest players.

Lemma 67. *For all randomness ω and all round r, consider two executions $\mathsf{REAL}(\omega)$ and $\mathsf{HYB}^r(\omega)$. Consider an honest player P at round r_1, where $r_1 > r$. Let $\mathcal{C}$ be the best chain at round r_1 in the execution $\mathsf{REAL}(\omega)$, and let $\mathcal{C}_{\mathsf{hyb}}$ be the best chain at round r_1 in the execution $\mathsf{HYB}^r(\omega)$. Then, we have, $\mathsf{len}(\mathcal{C}) \geq \mathsf{len}(\mathcal{C}_{\mathsf{hyb}})$.*

Proof. As demonstrated in Lemma 66, the tiebreak rule described in Subsection 4.3.3 guarantees that honest players always follow the best chain with the fastest expected time for extension. Therefore, if the adversary broadcasts its chain to the honest players, the growth of the best chain will accelerate, making it impossible for the adversary to impede the chain growth of the honest players. □

To analyze the chain growth from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$, we consider the hybrid execution $\mathsf{HYB}^r(\omega)$. Since the first r rounds in the hybrid execution $\mathsf{HYB}^r(\omega)$ are the same as in the real execution $\mathsf{REAL}(\omega)$, the best chain at round r in both executions is the same. Moreover, as stated in Lemma 67, the best chain at round r_1 in the real execution $\mathsf{REAL}(\omega)$ is longer than the best chain in the hybrid execution $\mathsf{HYB}^r(\omega)$. Thus, the chain growth from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$ is larger than the chain growth in the hybrid execution $\mathsf{HYB}^r(\omega)$.

4.6.2 Analyzing the chain growth property via a simplified Markov chain

Next, we design Markov chains and a chain growth function to analyze the chain growth in the hybrid execution. In this subsection, we introduce a simplified Markov chain. Then, in Subsection 4.6.3, we extend the simplified Markov chain to design a

more complex augmented Markov chain. Using the augmented Markov chain, we can obtain a tighter bound for chain growth.

We will use the definition of the *depth-based subsets* to design the simplified Markov chain. We show how to use the depth-based subsets to analyze the chain growth in Subsection 4.6.2.1. Here, each state in the simplified Markov chain contains information about the number of chains in the depth-based subsets. Then, we present a simplified Markov chain to analyze the chain growth property for $D = 1$ in Subsection 4.6.2.2. Finally, we present a simplified Markov chain to analyze the chain growth property for an arbitrary D in Subsection 4.6.2.3.

4.6.2.1 Depth-based subsets in the set of best chains in the execution

We represent the definition of depth-based subsets in the set of best chains. Consider a protocol execution at a certain round, let $\mathcal{C}_{\text{best}}$ denote the best chain and $\mathbb{C}_{\text{best}}$ be the set of best chains. We partition the set $\mathbb{C}_{\text{best}}$ into $D+1$ number of *disjoint subsets* based on the length of those chains. Let $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$ be the length of the best chain, and for all $i \in [0..D]$, the i-depth-based subset L_i is the subset of chains with the length of $\ell - i$ in the set $\mathbb{C}_{\text{best}}$. That is, $L_i = \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$.

Let $n_0 = |L_0|$ be the number of chains in 0-depth subset. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0) \approx n_0 \cdot \alpha$ (see Figure 25). Recall that, if the honest players follows the single extension framework and only extend the best chain, then the probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(1) = \alpha$. Here, the amplification ratio $\hat{\mathtt{A}}^{\bullet}_{D}$ can be estimated by the average number of chains in L_0. More concretely, let $\mathbb{E}[\mathsf{w}(n_0)]$ be the expected value of the number of chains n_0 in L_0 and $\mathbb{E}[\mathsf{w}(n_0)]$ be the expected value of the probability that the honest players generate a new best chain. We have,

the amplification ratio is

$$\hat{\mathtt{A}}^{\bullet}_{D} = \frac{\mathbb{E}[\mathsf{w}(n_0)]}{\alpha} \approx \frac{\mathbb{E}[n_0 \cdot \alpha]}{\alpha} = \mathbb{E}[n_0].$$

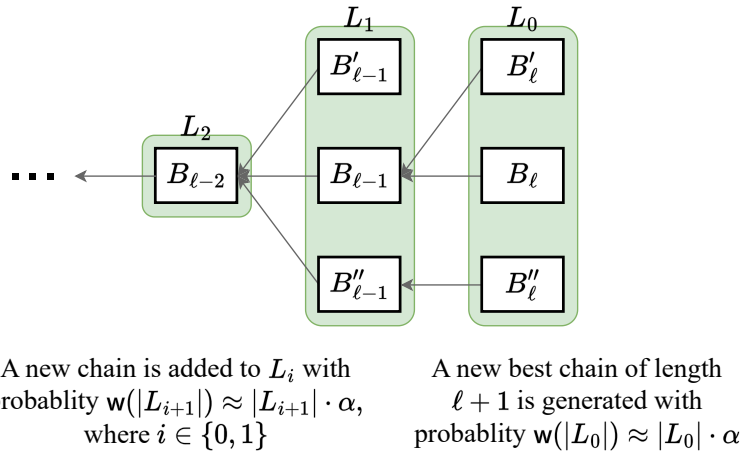


Figure 25: Partitioning a set of best chains $\mathbb{C}_{\text{best}}$ into multiple disjoint depth-based subsets for $D = 2$. Note that, the set of best chains $\mathbb{C}_{\text{best}}$ here is identical to the one in Figure 22. The set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into 3 subsets L_0, L_1, L_2. Let ℓ be the length of the best chain. Here, the probability that honest players generate a new best chain of length $\ell+1$ is $\mathsf{w}(|L_0|)$. The probability that honest players generate a new chain in L_0 is $\mathsf{w}(|L_1|)$. The probability that honest players generate a new chain in L_1 is $\mathsf{w}(|L_2|)$.

To analyze the number of chains in L_0, we need to analyze the number of chains in L_1. Similarly, for all $i \in [D-1]$, to analyze the number of chains in L_i, we need to analyze the number of chains in L_{i+1}. Thus, to compute the amplification ratio, we develop a Markov chain that consists of multiple states. The states in the Markov chain represent the number of chains in all depth-based subsets in the set of best chains. Note that, as the subset L_D always has exactly one chain, we omit the representation of $|L_D|$ in the Markov chain state.

4.6.2.2 The simplified Markov chain for $D = 1$

We describe the simplified Markov chain with the state space S and a transition matrix $\mathbf{T}$ to analyze the chain growth for $D = 1$. Here, each state in S is in the format of $\langle n_0 \rangle$ that represents a protocol round in which a set of best chains in which the number of chains in 0-depth subset is n_0, where $n_0 \in \mathbb{N}$. The state space is given as $S = \{\langle n_0 \rangle\}_{n_0 \in \mathbb{N}}$. More concretely, consider the set of best chains that is partitioned into two subsets L_0 and L_1. Let ℓ be the length of the current best chain. Let $n_0 = |L_0|$ and $n_1 = |L_1|$ be the numbers of chains in L_0 and L_1, respectively. Note that, the number of chains in L_1 always equals 1, i.e., $n_1 = 1$. Thus, we omit the representation of $|L_1|$ in the states of the Markov chain.

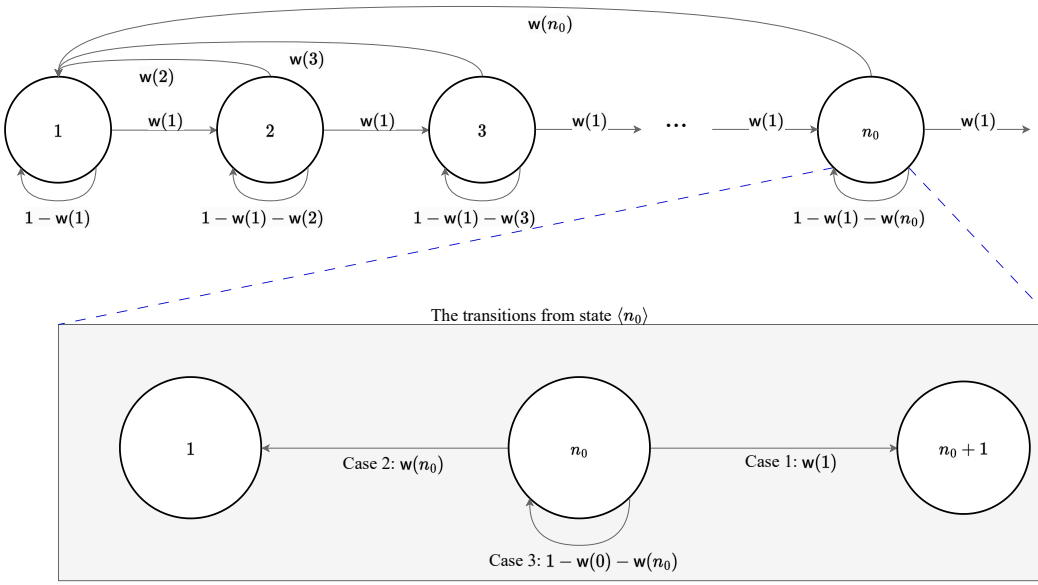


Figure 26: The complete state machine for the simplified Markov chain for $D = 1$.

The transition matrix $\mathbf{T}$ is an $|S| \times |S|$ matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\mathbf{T}$ as follows. Initially, we set all the values in $\mathbf{T}$ to 0. Then, for each state $\langle n_0 \rangle$, we update

the transition matrix based on the following cases of transitions (see Figure 26).

Case 1: A new chain of length ℓ is generated. In other words, a chain of length $\ell-1$ in L_1 is extended. As the number of chains in L_0 is one, the probability that the honest players generate a new chain of length ℓ is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle n_0 \rangle$ to state $\langle n_0 + 1 \rangle$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle n_0+1 \rangle} := \mathsf{w}(1)$.

Case 2: A new best chain of length $\ell + 1$ is generated. In other words, a chain of length ℓ in L_0 is extended. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in L_0. In this case, the new 0-depth subset of the new set of best chains only consists of one chain, i.e., the new best chain. The state machine moves from state $\langle n_0 \rangle$ to state $\langle 1 \rangle$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle 1 \rangle} := \mathsf{w}(n_0)$.

Case 3: No new chain is generated. The state machine remains at the current state $\langle n_0 \rangle$ with a probability of $1 - \mathsf{w}(1) - \mathsf{w}(n_0)$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle n_0 \rangle} := 1 - \mathsf{w}(1) - \mathsf{w}(n_0)$.

Let q_{n_0} be the stationary probability of the state $\langle n_0 \rangle$. Similar to Equation 4.1, for all $s \in S$, we have, $\sum_{s \in S} q_s = 1$ and $q_s = \sum_{s' \in S} q_{s'} \cdot \mathbf{T}_{s',s}$. We define the chain growth function $\mathsf{growth} : S \to [0, 1]$ such that $\mathsf{growth}(n_0) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}_1^\bullet$ is computed as the expected number of chains in L_0 in each round, i.e.,

$$\hat{\mathtt{A}}_1^\bullet = \sum_{n_0=1}^{\infty} \left(q_{n_0} \cdot \frac{\mathsf{growth}(n_0)}{\alpha} \right) = \sum_{n_0=1}^{\infty} q_{n_0} \cdot n_0.$$

Based on the equation, we have $\hat{\mathtt{A}}_1^\bullet = 1.39$.

4.6.2.3 The simplified Markov chain for a general D

We now describe the simplified Markov chain with the state space S and a transition matrix $\mathbf{T}$ to analyze the chain growth for arbitrary D. Consider a protocol round in which the set of best chains is $\mathbb{C}_{\text{best}}$. For $i \in [0..D]$, let L_i be the

i-depth subset in $\mathbb{C}_{\text{best}}$ and $n_i = |L_i|$ be the number of chains in L_i. In the simplified Markov chain, we use the state $\langle n_0, \cdots, n_{D-1} \rangle$ to represent the protocol round with the set of best chains $\mathbb{C}_{\text{best}}$. Note that, since the subset L_D always has exactly one chain, i.e., $n_D = 1$, we does not include the number of chains in L_D in the representation of the states. The state space of the simplified Markov chain is given as $S = \{\langle n_0, \cdots, n_{D-1} \rangle\}_{n_i \in \mathbb{N}, \forall i \in [0..D-1]}$.

The state of the simplified Markov chain provides information about the set of best chains in each round. After each round, the state machine transitions to a new state depending on updates in the set of best chains. We categorize the transitions in the simplified Markov chain based on how the set of the best chains is updated as follows.

- *Case 1: A new chain is generated but the length of the best chain remains the same.* In this case, a chain will be added to a depth-based subset. The remaining depth-based subsets will remain the same. The state machine moves to a new state in which the number of chains in the updated depth-based subset increases by one.

- *Case 2: A new best chain is generated, i.e., the length of the best chain increases by* 1. As a result, the new best chain is added to the set of best chains, while some existing chains may be removed. Based on the number of chains in the depth-based subsets, we cannot know how many chains are removed. Thus, we consider the worst-case scenario where the set of best chains only consists of the best chain and its prefixes. The state machine moves to the new state in which the number of each depth-based subset contains only one block.

- *Case 3: No chain is generated.* The set of best chains remains unchanged. The state machine remains at the same state.

The transition matrix $\mathbf{T}$ is constructed as follows. First, all values in $\mathbf{T}$ are initially set to 0. Then, for each state $\langle n_0, \cdots, n_{D-1} \rangle \in \mathbf{T}$, the transition matrix is updated based on the three cases of transitions (see Figure 27).

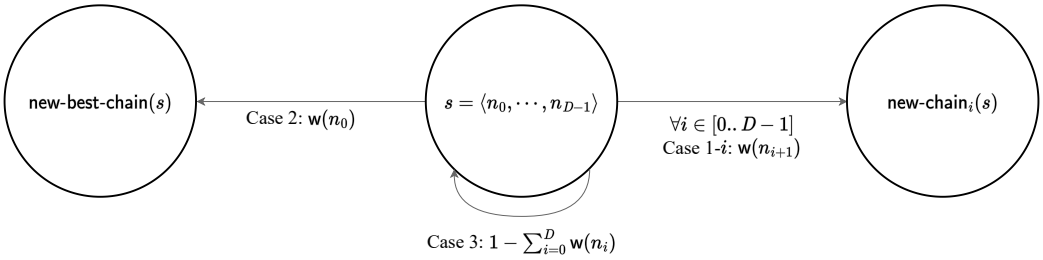


Figure 27: The transitions from state $s = \langle n_0, \cdots, n_{D-1} \rangle$ in the state machine of a simplified Markov chain for a general D.

Case 1: A new chain is generated but the length of the best chain remains the same. Here, a player generates a chain $\mathcal{C}$ such that $\mathsf{len}(\mathcal{C}) \leq \ell$. The chain $\mathcal{C}$ can be added to one of the i-depth subsets L_i in the set of best chains $\mathbb{C}_{\text{best}}$, where $i \in [0..D-1]$. This case is the generalization of the Case 1 in Figure 26. Based on the depth of the new chain, we divide this case into D sub-cases as follows. For any $i \in [0..D-1]$, we consider the sub-case 1-i in which the new chain is added to subset L_i. In other words, a chain in subset L_{i+1} is extended. The probability of such an event is $\mathsf{w}(n_{i+1})$, where n_{i+1} is the number of chains in the subset L_{i+1}. For each $i \in [0..D-1]$, we define the function $\mathsf{new\text{-}chain}_i : S \to S$ that takes as input a state in S and outputs an updated state when a new chain is added to the i-depths subset L_i. In other words, if a new chain is added to L_i, the state machine moves from state $\langle n_0, \cdots, n_{D-1} \rangle$ to state $\mathsf{new\text{-}chain}_i(n_0, \cdots, n_{D-1})$. Next, we show the definition of function $\mathsf{new\text{-}chain}_i$. For a state $\langle n_0, \cdots, n_{D-1} \rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1} \rangle = \mathsf{new\text{-}chain}_i(n_0, \cdots, n_{D-1})$. We have, $n'_{i'} = n_{i'}$, $\forall i' \neq i$ and $n'_{i'} = n_{i'} + 1$ if $i' = i$. For example, with $D = 1$, we have, $\mathsf{new\text{-}chain}_0(n_0) = \langle n_0 + 1 \rangle$. We set $\mathbf{T}_{\langle n_0, \cdots, n_{D-1} \rangle, \mathsf{new\text{-}chain}_i(n_0, \cdots, n_{D-1})} := \mathsf{w}(n_{i+1})$.

Case 2: A new best chain is generated, i.e., the length of the best chain increases by 1. In this case, a chain in the subset L_0 is extended. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in the subset L_0. This case is the generalization of the Case 2 in Figure 26. In this case, we consider the worst-case scenario where the set of best chains only consists of the best chain and its prefixes, meaning each depth-based subset contains only one block. This is because based on the information on depth-based subsets, we cannot determine how many chains will be removed from the set of best chains if the distance from the new best chains to those chains is greater than D. We define the function $\mathsf{new\text{-}best\text{-}chain} : S \to S$ that takes as input a state in S and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, the state machine moves from state $\langle n_0, \cdots, n_{D-1} \rangle$ to state $\mathsf{new\text{-}best\text{-}chain}(n_0, \cdots, n_{D-1})$. Next, we show the definition of function $\mathsf{new\text{-}best\text{-}chain}$. For a state $\langle n_0, \cdots, n_{D-1} \rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1} \rangle = \mathsf{new\text{-}best\text{-}chain}(n_0, \cdots, n_{D-1})$. We have, $n'_{i'} = 1$, $\forall i' \in [0..D-1]$. For example, with $D = 1$, we have, $\mathsf{new\text{-}best\text{-}chain}(n_0) = \langle 1 \rangle$. We set $\mathbf{T}_{\langle n_0, \cdots, n_{D-1} \rangle, \mathsf{new\text{-}best\text{-}chain}(n_0, \cdots, n_{D-1})} := \mathsf{w}(n_0)$.

Case 3: No chain is generated. In this case, the state machine remains at the current state $\langle n_0, \cdots, n_{D-1} \rangle$ with a probability of $1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$. We set $\mathbf{T}_{\langle n_0, \cdots, n_{D-1} \rangle, \langle n_0, \cdots, n_{D-1} \rangle} := 1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$.

Let $q_{n_0, \cdots, n_{D-1}}$ be the stationary probability of the state $\langle n_0, \cdots, n_{D-1} \rangle$. Similar to Equation 4.1, we have,

$$\begin{cases} \sum_{n_1=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} q_{n_0, \cdots, n_{D-1}} = 1, \\ q_{n_0, \cdots, n_{D-1}} = \sum_{\langle n'_0, \cdots, n'_{D-1} \rangle \in S} \left(q_{n'_0, \cdots, n'_{D-1}} \cdot \mathbf{T}_{\langle n'_0, \cdots, n'_{D-1} \rangle, \langle n_0, \cdots, n_{D-1} \rangle} \right). \end{cases} \tag{4.3}$$

We define the chain growth function $\mathsf{growth} : S \to [0, 1]$ such that $\mathsf{growth}(n_0, \cdots, n_{D-1}) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}^{\bullet}_{D}$ equals the expected number of chains in L_0 in each round, i.e.,

$$\begin{aligned}\hat{\mathtt{A}}^{\bullet}_{D} &= \sum_{n_0=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} \left(q_{n_0,\cdots,n_{D-1}} \cdot \frac{\mathsf{growth}(n_0, \cdots, n_{D-1})}{\alpha} \right) \\ &= \sum_{n_0=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} \left(q_{n_0,\cdots,n_{D-1}} \cdot n_0 \right).\end{aligned}$$

Using the simplified Markov chain, we can find a lower bound of the amplification ratio as shown in Figure 23. For $D = 50$, we can find a lower bound $\hat{\mathtt{A}}^{\bullet}_{50} \geq 1.56$.

4.6.3 Analyzing the chain growth via an augmented Markov chain

In the simplified Markov chain, the state only provide the information on the number of chain in the depth-based subsets. Some critical information is omitted due to this simple representation. Indeed, when a new best chain is generated and some of the chains are removed from the set of best chains, we have to consider the worst-case scenario. Hence, we cannot establish a good lower bound of the amplification ratio, *even when D is big!* For example, with $D = 50$, using the simplified Markov chain, we can only guarantee an amplification of 1.56. Here, we present an **augmented** Markov chain in which *each state contains more information on the set of best chains*. With the augmented Markov chain, we can find a *better* lower bound of the amplification ratio.

Before presenting the augmented Markov chain, we introduce the notion of *depth-distance-based subsets* in Subsection 4.6.3.1. The chains in a depth-distance-based subset are selected based on *both* the length of those chains *and* the distance from the best chain to those chains. Each state in the augmented Markov chain contains information about the number of chains in each depth-distance-based subset. With

this information, we can determine the number of chains removed from the set of best chains when a new longest chain is generated. This avoids having to consider the worst-case scenario where each depth-based subset contains only one chain, as in the simplified Markov chain.

We first present a augmented Markov chain to analyze the chain growth for $D = 2$ in Subsection 4.6.3.2 and then extend the augmented Markov chain for an arbitrary D in Subsection 4.6.3.3.

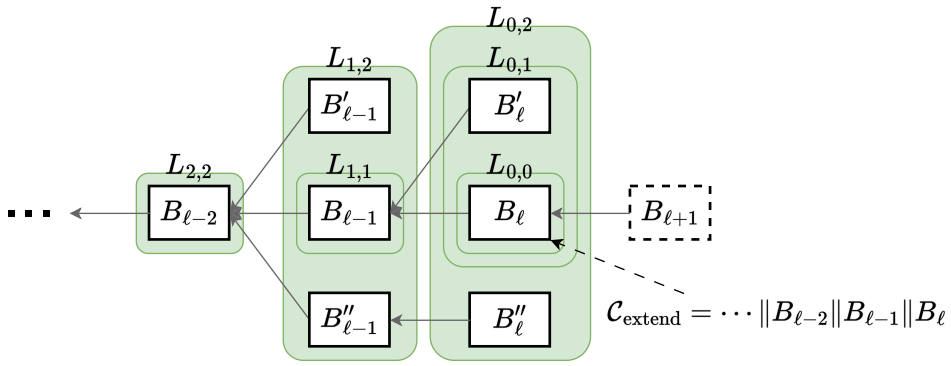


Figure 28: The depth-distance-based subsets of the set of best chains $\mathbb{C}_{\text{best}}$ for $D = 2$. Recall that, in Figure 25, the set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into 3 disjoint depth-based subsets $L_0, L_1, \cdots, L_2$. Let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. In this figure, $\mathcal{C}_{\text{extend}} = \cdots \| B_{\ell-1} \| B_{\ell-1} \| B_{\ell}$. (In some future round, a new best chain is generated by adding a block $B_{\ell+1}$ to $\mathcal{C}_{\text{extend}}$.) Consider a i-depth subset L_i, where $i \in [0..D]$. We further define multiple subsets of chains based on the distance from $\mathcal{C}_{\text{extend}}$ to those chains in L_i. More concretely, for $i \in [0..D]$, $j \in [i..D]$, the "i-depth j-distance" subset $L_{i,j}$ consists of all the chains in the i-depth subset L_i such that the distance from the chain $\mathcal{C}_{\text{extend}}$ to those chains does not exceed j, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq j\}$.

4.6.3.1 Depth-distance-based subsets in the set of best chains in the execution

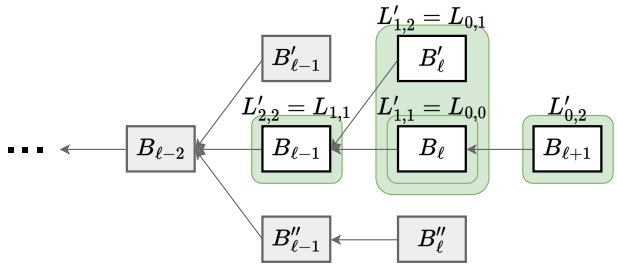


Figure 29: The new set of best chains $\mathbb{C}'_{\text{best}}$ when a new block is added on $\mathcal{C}_{\text{extend}}$. Here, the set of best chains $\mathbb{C}'_{\text{best}}$ consists of the chains in which the last blocks of those chains are $B_{\ell-1}, B_{\ell}, B'_{\ell}, B_{\ell+1}$. The chains in which the last block of those chains are $B_{\ell-2}, B'_{\ell-1}, B''_{\ell-1}, B''_{\ell}$ are belong to $\mathbb{C}_{\text{best}}$ but not $\mathbb{C}'_{\text{best}}$. For $i \in [0..D], j \in [i..D]$, let $L'_{i,j}$ be the "i-depth j-distance" subset of the set of best chains $\mathbb{C}'_{\text{best}}$, i.e., $L'_i = \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$. The subset L'_0 only consists of the best chain $\mathcal{C}'_{\text{best}} = \cdots \| B_{\ell-2} \| B_{\ell-1} \| B_{\ell} \| B_{\ell+1}$. For $i \in [1..D-1]$, the i-depth subset of $\mathbb{C}'_{\text{best}}$ can be obtained by the depth-distance-based subsets of $\mathbb{C}_{\text{best}}$ in Figure 28. Indeed, $L'_{i,j} = L_{i-1,j-1}$, where $L_{i+1,D-1}$ is the "$(i-1)$-depth $(D-1)$-distance" subset of $\mathbb{C}_{\text{best}}$.

We introduce the notion of depth-distance-based subsets. Let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. We remark that, the chain $\mathcal{C}_{\text{extend}}$ can be changed when the set of best chain $\mathbb{C}_{\text{best}}$ is updated. Indeed, when the a chain $\mathcal{C}'$ is added to subset L_0, the chain $\mathcal{C}'$ can be extended earlier than the current chain $\mathcal{C}_{\text{extend}}$. In this case we update $\mathcal{C}_{\text{extend}} = \mathcal{C}'$. For $i \in [0..D]$, we define multiple subset of the i-depth subset L_i as follows. For $j \in [0..D]$, the "i-depth j-distance" subset $L_{i,j}$ is the set of chains in the i-subset L_i such that the distance from $\mathcal{C}_{\text{extend}}$ to those chains does not exceed

j, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq j\}$. Here, for all $j < i$, $L_{i,j} = \emptyset$ and $L_{i,i}$ consists of exact 1 chain, i.e., the prefix of $\mathcal{C}_{\text{extend}}$ with the length $\ell - i$, where $\ell = \mathsf{len}(\mathcal{C}_{\text{extend}})$. Further, $L_{i,j}$ is a subset of $L_{i,j+1}$, i.e., $L_{i,j} \subseteq L_{i,j+1}$. In other words, $L_{i,i} \subseteq L_{i,i+1} \subseteq \cdots \subseteq L_{i,D}$. For example, in Figure 28, the "0-depth 0-distance" $L_{0,0}$ consists of 1 chain that has the last block is B_ℓ. The "0-depth 1-distance" $L_{0,1}$ consists of 2 chains that have the last blocks are B_ℓ, B'_ℓ. The "0-depth 2-distance" $L_{0,2}$ consists of 3 chains that have the last blocks are $B_\ell, B'_\ell, B''_\ell$.

When the chain $\mathcal{C}_{\text{extend}}$ is extended, a new best chain is added and some chains are removed form the set of best chain (see Figure 29 for an example). Let $\mathcal{C}'_{\text{best}}$ be the new best chain and $\mathbb{C}'_{\text{best}}$ be the new set of best chain. Let L'_i be the i-depth subset of the set of best chains $\mathbb{C}'_{\text{best}}$, i.e., $L'_i = \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$. The subset L'_0 only consists of the best chain $\mathcal{C}'_{\text{best}} = \cdots \| B_0 \| B_2 \| B_5 \| B_8 \| B_{10}$. For $i \in [1..D]$, the i-depth subset of $\mathbb{C}'_{\text{best}}$ can be obtained by the depth-distance-based subsets of $\mathbb{C}_{\text{best}}$ in Figure 28. Indeed, for any chain $\mathcal{C}$, we have, $\mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) = \mathsf{distance}(\mathcal{C}'_{\text{best}} \to \mathcal{C}) - 1$. Thus, $\mathcal{C}'_{\text{best}} = \{\mathcal{C} : \mathsf{distance}(\mathcal{C}'_{\text{best}} \to \mathcal{C}) \leq D\} = \{\mathcal{C} : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq D - 1\}$. Hence, $L'_i = \{\mathcal{C} : \mathsf{len}(\mathcal{C}) = \ell + 1 - i \wedge \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq D - 1\} = L_{i-1,D-1}$.

4.6.3.2 The augmented Markov chain for $D = 2$

We will now design an augmented Markov chain to analyze chain growth. The augmented Markov chain keeps track of the number of chains in $L_{i,j}$ for different values of i and j, providing a way to analyze chain growth. The states in the augmented Markov chain capture information about the number of chains in each $L_{i,j}$, where $i \in [0..D]$ and $j \in [0..D]$. Note that for all $i \in [1..D]$, we have $|L_{i,j}| = 0$ for $j \in [0..D - i - 1]$ and $|L_{i,D-i}| = 1$. Thus, for $i \in [1..D]$ and $j \in [0..D - i]$, we omit the representation of $L_{i,j}$.

For $D = 1$, the augmented Markov chain is equivalent to the simplified Markov

chain. Therefore, we will first design an augmented Markov chain for $D = 2$. Then, we will extend the design for a general D.

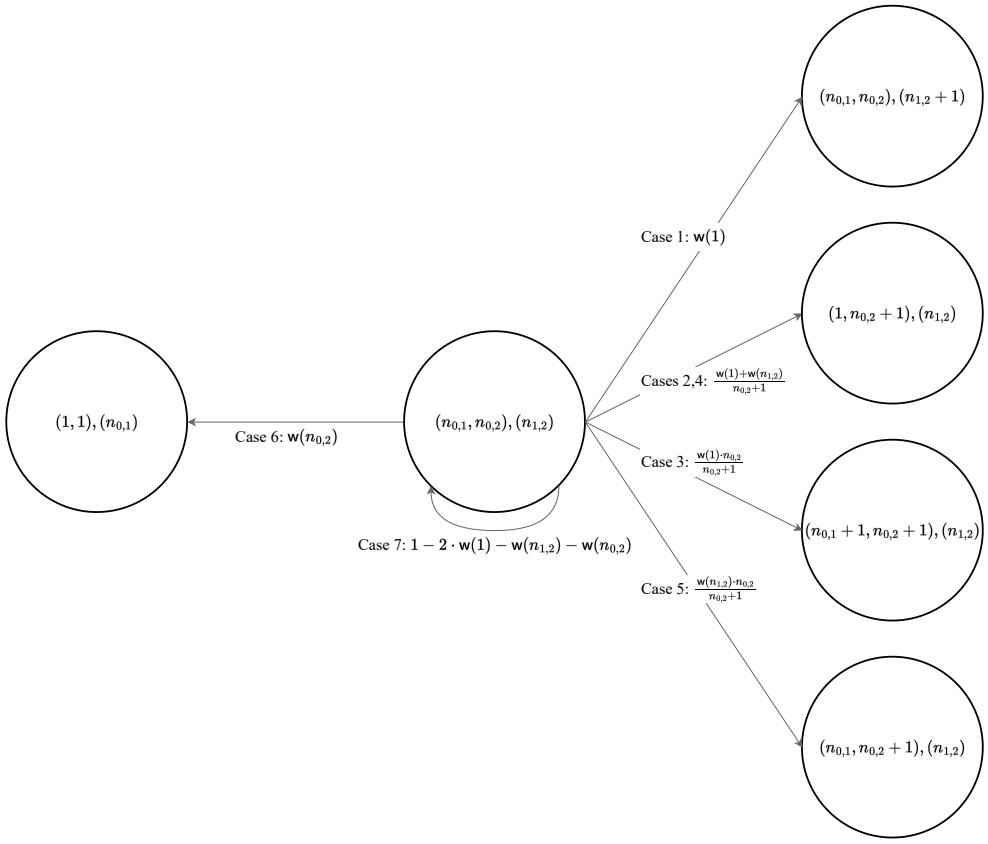


Figure 30: The transitions from a state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ in the state machine for $D = 2$.

We describe the augmented Markov chain with the state space $\hat{S}$ and a transition matrix $\hat{\mathbf{T}}$ to analyze the chain growth for $D = 2$. Consider a round with the set of best chains $\mathbb{C}_{\text{best}}$, let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. For $i, j \in [0..2]$, let $L_{i,j}$ be the "i-depth j-distance" subset of the set of best chain $\mathbb{C}_{\text{best}}$, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq j\}$. Let $n_{i,j} = |L_{i,j}|$ be the numbers of chains in $L_{i,j}$. In the augmented Markov chain, the state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ represents the protocol round with the set of best chains $\mathbb{C}_{\text{best}}$. (Note that, for $i \in [1..D]$

and $j \in [0..D-i]$, we omit the representation of $L_{i,j}$ since the numbers of chains in those subsets are the same for every state.) The state space of the augmented Markov is given as $\hat{S} = \{\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle\}_{n_{0,1},n_{0,2},n_{1,2} \in \mathbb{N}}$. Here, each state in $\hat{S}$ is in the format of $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ that represents a set of best chains such that the numbers of chains in "0-depth 1-distance", "0-depth 2-distance", and "1-depth 2-distance" subsets are $n_{0,1}, n_{0,2}, n_{1,2}$, respectively. The state space is given as $\hat{S} = \{\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle\}_{n_{0,1},n_{0,2},n_{1,2} \in \mathbb{N}}$. More concretely, consider the set of best chains and let $L_{i,j}$ $(i, j \in [0..D])$ be the i-depth j-distance subset of the set of best chains. Let $n_{i,j} = |L_{i,j}|$ be the numbers of chains in $L_{i,j}$. As we mentioned above, for $i \in [1..D]$ and $j \in [0..D-i]$, we omit the representation of $L_{i,j}$.

Recall that, the transition matrix $\hat{\mathbf{T}}$ is an $|\hat{S}| \times |\hat{S}|$ matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\hat{\mathbf{T}}$ as follows. Initially, we set all the values in $\hat{\mathbf{T}}$ to 0. Then, for each state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$, we update the transition matrix based on the following cases of transitions (see Figure 30).

Case 1: A new chain $\mathcal{C}$ is added to subset $L_{1,2}$. A chain $\mathcal{C}$ is added to subset $L_{1,2}$ if a chain in $L_{2,2}$ is extended. As the number of chains in the subset $L_{2,2}$ is 1, the probability of this case is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}+1)\rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1},n_{0,2}),(n_{1,2})\rangle, \langle (n_{0,1},n_{0,2}),(n_{1,2}+1)\rangle} := \mathsf{w}(1)$.

Case 2: A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0. A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ if a chain in subset $L_{1,1}$ is extended. As the number of chains in the subset $L_{1,1}$ is 1, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. The probability of this event

is $\frac{1}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(1, n_{0,2}+1), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(1,n_{0,2}+1),(n_{1,2})\rangle} := \frac{\mathsf{w}(1)}{n_{0,2}+1}$.

Case 3: A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0. As we show in Case 2, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(n_{0,1}+1, n_{0,2}+1), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(n_{0,1}+1,n_{0,2}+1),(n_{1,2})\rangle} := \frac{n_{0,2}\cdot\mathsf{w}(1)}{n_{0,2}+1}$.

Case 4: A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0. A new chain is added to subset $L_{0,2}$ if a chain in the subset $L_{1,2}$ is extended. As the number of chains in the subset $L_{1,2}$ is $n_{1,2}$, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(1, n_{0,2}+1), (n_{1,2})\rangle$. Recall that, we already added a transition from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(1, n_{0,2}+1), (n_{1,2})\rangle$ in Case 2. Thus, we set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(1,n_{0,2}+1),(n_{1,2})\rangle} := \frac{\mathsf{w}(n_{1,2})}{n_{0,2}+1} + \frac{\mathsf{w}(1)}{n_{0,2}+1}$.

Case 5: A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_2. As we show in Case 4, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(n_{0,1}, n_{0,2}+1), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(n_{0,1},n_{0,2}+1),(n_{1,2})\rangle} := \frac{n_{0,2}\cdot\mathsf{w}(n_{1,2})}{n_{0,2}+1}$.

Case 6: A new best chain is generated, i.e., the length of the best chain increases by 1. A new best chain is generated if a chain in the subset L_0 is extended. As $L_0 = L_{0,2}$, the

number of chains in L_0 is $n_{0,2}$. Thus, with a probability of $\mathsf{w}(n_{0,2})$, a new best chain is generated. After the new chain is generated, the 0-depth subset only consists of the best chain. Thus, the number of chains in "0-depth 1-distance" subset and "0-depth 2-distance" subset are the same and equal 1. The number of chains in "1-depth 2-distance" subset is $n_{0,1}$. Thus, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ to state $\langle (1,1), (n_{0,1}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1},n_{0,2}),(n_{1,2}) \rangle, \langle (1,1),(n_{0,1}) \rangle} := \mathsf{w}(n_{0,2})$.

Case 7: No new chain is generated. With a probability of $1 - 2 \cdot \mathsf{w}(1) - \mathsf{w}(n_{0,2}) - \mathsf{w}(n_{0,2})$, the state machine remains at the current state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1},n_{0,2}),(n_{1,2}) \rangle, \langle (n_{0,1},n_{0,2}),(n_{1,2}) \rangle} := 1 - 2 \cdot \mathsf{w}(1) - \mathsf{w}(n_{0,2}) - \mathsf{w}(n_{0,2})$.

Let $q_{(n_{0,1},n_{0,2}),(n_{1,2})}$ be the stationary probability of the state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$. Similar to Equation 4.1, we have,

$$\begin{cases} \sum_{s \in \hat{S}} q_s = 1, \\ q_s = \sum_{s' \in \hat{S}} q_{s'} \cdot \hat{\mathbf{T}}_{s',s}, \forall s \in \hat{S}. \end{cases} \tag{4.4}$$

The equations in Equation 4.4 equivalent to the following.

$$\begin{cases} \sum_{n_{0,1}=1}^{\infty} \sum_{n_{0,2}=1}^{\infty} \sum_{n_{1,2}=1}^{\infty} q_{(n_{0,1},n_{0,2}),(n_{1,2})} = 1, \\ 3 \cdot \mathsf{w}(1) \cdot q_{(1,1),(n_{1,2})} = \sum_{n_{2,1}=1}^{\infty} \sum_{n_{2,2}=1}^{\infty} q_{(n_{0,1},n_{0,2}),(n_{1,2})}, \\ (2 \cdot \mathsf{w}(1) + \mathsf{w}(n_{0,2})) \cdot q_{(n_{0,1},n_{0,2}),(n_{1,2})} = q_{(n_{0,1}-1,n_{0,2}-1),(n_{1,2})} \cdot \mathsf{w}(1), \\ (\mathsf{w}(1) + \mathsf{w}(n_{1,2}) + \mathsf{w}(n_{0,1})) \cdot q_{(1,n_{0,2}),(n_{1,2})} = q_{(n_{0,1},n_{0,2})-1,(n_{1,2})} \cdot \mathsf{w}(1) + q_{(n_{0,1},n_{0,2}),(n_{1,2})} \cdot \mathsf{w}(n_1). \end{cases} \tag{4.5}$$

We define the chain growth function $\mathsf{growth} : \hat{S} \to [0,1]$ such that $\mathsf{growth}((n_{0,1}, n_{0,2}), (n_{1,2})) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}_2^{\bullet}$ equals the expected number of chains in

L_0 in each round, i.e.,

$$\hat{\mathtt{A}}_2^{\bullet} = \sum_{n_{0,1}=1}^{\infty} \sum_{n_{0,2}=1}^{\infty} \sum_{n_{1,2}=1}^{\infty} \left(q_{(n_{0,1},n_{0,2}),(n_{1,2})} \cdot \frac{\mathsf{growth}((n_{0,1}, n_{0,2}), (n_{1,2}))}{\alpha} \right)$$
$$= \sum_{n_{0,1}=1}^{\infty} \sum_{n_{0,2}=1}^{\infty} \sum_{n_{1,2}=1}^{\infty} \left(q_{(n_{0,1},n_{0,2}),(n_{1,2})} \cdot n_{0,2} \right).$$

Combining with Equation 4.5, we have, $\hat{\mathtt{A}}_2^{\bullet} \approx 1.51$.

4.6.3.3 The augmented Markov chain for a general D

We now describe the augmented Markov chain with the state space $\hat{S}$ and a transition matrix $\hat{\mathbf{T}}$ to analyze the chain growth for $D = 2$. Consider a round with the set of best chains $\mathbb{C}_{\text{best}}$. Let L_i be the i-depth subset in $\mathbb{C}_{\text{best}}$, where $i \in [0..D]$. Let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. For $i \in [0..D]$ and $j \in [0..D]$, let $L_{i,j}$ be the "i-depth j-distance" subset of the set of best chains $\mathbb{C}_{\text{best}}$, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq j\}$. Let $n_{i,j} = |L_{i,j}|$ be the number of chains in $L_{i,j}$. In the augmented Markov chain, the state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ represents the protocol round with the set of best chains $\mathbb{C}_{\text{best}}$. (We remark that, for $i \in [1..D]$ and $j \in [0..D-i]$, we do not include the number of chains in $L_{i,j}$ since they are the same for every state.) Hence, the state space of the augmented Markov chain is given as $\hat{S} = \{\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle\}_{n_{i,j} \in \mathbb{N}, \forall i \in [0..D-1], j \in [i+1..D]}$.

Similar to the simplified Markov chain, the state of the augmented Markov chain represents the information about the set of best chains in each round. We categorize the transitions in the simplified Markov chain based on how the set of the best chains is updated as follows.

- *Case 1: A new chain that is shorter than the current best chain is generated.* In this case, a chain will be added to a distance-depth-based subset and all its supersets. The remaining distance-depth-based subsets will remain the same.

As a result, the state machine will transition to a new state in which the number of chains in the updated depth-based subsets has increased by one.

- Case 2: *A new chain that has the same length as the current best chain is generated and added to L_0, and the new chain is not the first chain in subset L_0 that is extended.* In this case, the updates on the set of best chains are similar to Case 1. A chain will be added to a distance-depth-based subset and its supersets that are subsets of the 0-depth subset. The state machine moves to the new state in which the number of chains in the updated depth-based subsets increases by one.

- Case 3: *A new chain that has the same length as the current best chain is generated and added to L_0, and the new chain is the first chain in subset L_0 that is extended.* In this case, the new chain is added to the set of best chains without removing any chains. Additionally, the chain $\mathcal{C}_{\text{extend}}$ is updated as the new chain. The state machine transitions to a new state in which the number of chains in distance-depth-based subsets is updated based on the new value of $\mathcal{C}_{\text{extend}}$.

- Case 4:*A new best chain is generated, i.e., the length of the best chain increase by* 1. The new best chain has been added and some of the chains have been removed from the set of best chains. The chain $\mathcal{C}_{\text{extend}}$ has been updated to become the new best chain. The state machine has moved to a new state where the number of chains in distance-depth-based subsets is determined based on the number of chains in distance-depth-based subsets in the current state.

- Case 5: *No chain is generated.* The set of best chains remains unchanged. The state machine remains at the same state.

Let $\hat{\mathbf{T}}$ be an $|\hat{S}| \times |\hat{S}|$ transition matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\hat{\mathbf{T}}$ as follows. Initially, we set all the values in $\hat{\mathbf{T}}$ to 0. Then, for each state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$, we update the transition matrix based on the following cases of transitions (see Figure 31).

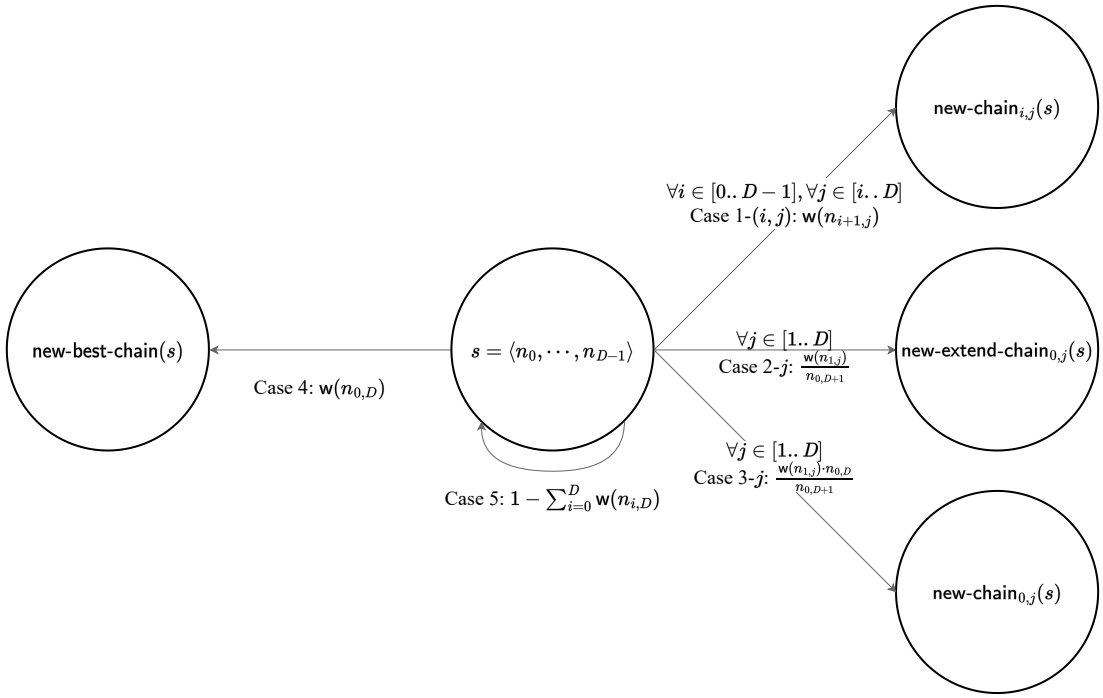


Figure 31: The transition from state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ in the state machine for a general D.

Case 1: A new chain $\mathcal{C}$ that is shorter than the current best chain is generated. Based on the depth and the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider the following sub-cases. For $i \in [1..D-1], j \in [i+1..D]$, we consider the sub-case 1-(i,j) where the new chain $\mathcal{C}$ is added to "i-depth j-distance" subset $L_{i,j}$. In other words, a player extends a chain in "$(i+1)$-depth j-distance" subset $L_{i+1,j}$. As the number of chains in $L_{i+1,j}$ is $n_{i+1,j}$, the probability of such event is

$\mathsf{w}(n_{i+1,j})$. This case is generalized from Case 1 in Figure 30. We define a function $\mathsf{new\text{-}chain}_{i,j} : \hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain is added to the subset $L_{i,j}$. In other words, if a new chain is added to $L_{i,j}$, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}chain}_{i,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$.

Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}chain}_{i,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. We have, $n'_{i',j'} := n_{i',j'}$ for all $i' \neq i, j' \in [i'+1..D]$ or $i' = i, j' \in [i+1..j-1]$; and $n'_{i,j'} := n_{i,j'} + 1$ for all $j' \in [j..D]$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}chain}_{i,j}(s)} := \mathsf{w}(n_{i+1,j})$.

Case 2: A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is not the first chain in subset L_0 that is extended. Based on the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider D sub-cases as follows. For $j \in [1..D]$, we consider a sub-case 2-j in which the new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$. In other words, a player extends a chain in "1-depth j-distance" subset $L_{1,j}$. As the number of chains in the subset $L_{1,j}$ is $n_{1,j}$, the probability of such event is $\mathsf{w}(n_{1,j})$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is not the first chain in subset L_0 that is extended is $\frac{n_{0,D}}{n_{0,D}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, n_{i,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}chain}_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. Here, function $\mathsf{new\text{-}chain}_{0,j}$ is defined as in Case 1. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}chain}_{0,j}\ (\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle)$. For all $i' \neq 0, j' \in [i'+1..D]$ or $i' = 0, j' \in [i+1..j-1]$, we have, $n'_{i',j'} := n_{i',j'}$; and for all $j' \in [j..D]$, we have, $n'_{0,j'} := n_{0,j'} + 1$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}chain}_{0,j}(s)} := \frac{\mathsf{w}(n_{1,j}) \cdot n_{0,D}}{n_{0,D}+1}$.

Case 3: A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is the first chain in subset L_0 that is extended. Similar to case 2, for $j \in [1..D]$, we consider a sub-case 2-j in which the new chain $\mathcal{C}$ is

added to "0-depth j-distance" subset $L_{0,j}$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is the first chain in subset L_0 that is extended is $\frac{1}{n_{0,D}+1}$. We define a function $\mathsf{new\text{-}extend\text{-}chain}_{0,j} : \hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain $\mathcal{C}$ is added to the "0-depth j-distance" subset $L_{i,j}$ and the chain $\mathcal{C}$ is the first chain in subset L_0 that is extended. If such event happens, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}extend\text{-}chain}_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}extend\text{-}chain}_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. We have, $n'_{i',j'} := 1$ for all $i' \in [0..D], j' \in [i'+1..D-1]$; $n'_{i',D} := n_{i',D}$ for all $i' \in [0..D]$, and $n'_{0,D} := n_{0,D} + 1$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}extend\text{-}chain}_{0,j}(s)} := \frac{\mathsf{w}(n_{1,j})}{n_{0,D}+1}$.

Case 4: A new best chain is generated, i.e., the length of the best chain increase by 1. This case is a generalization of Case 4 in Figure 30. A new best chain is generated if a chain in the 0-depth subset L_0 is extended. As the number of chains in the subset L_0 is $n_{0,D}$, the probability of this event is $\mathsf{w}(n_{0,D})$. We define the function $\mathsf{new\text{-}best\text{-}chain} : \hat{S} \to \hat{S}$ that takes a state in $\hat{S}$ as input and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, The state machine moves from state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}best\text{-}chain}(s)$. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}best\text{-}chain}(s)$. We have, $n'_{0,j'} := 1$ for all $j' \in [1..D]$; and $n'_{i',j'} := n_{i'+1,j'-1}$ for all $i \in [1..D-1], j' \in [i'+1..D]$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}best\text{-}chain}(s)} := \mathsf{w}(n_{0,D})$.

Case 5: No new chain is generated. This case is generalized from Case 5 in Figure 30. The probability of this event is $1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$. Here, the state machine remains at the same state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle, \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle} := 1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$.

Let $q_{(n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})}$ be the stationary probability of the state $\langle (n_{0,1}, \cdots, n_{0,D}),$

$\cdots, (n_{D-1,D})\rangle$. Based on Equation 4.1, we have,

$$\begin{cases} \sum_{n_{0,1}=1}^{\infty} \cdots \left(\sum_{n_{0,D}=1}^{\infty} \cdots \left(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \right) \right) = 1, \\ q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} = \sum_{\langle (n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})\rangle \in \hat{S}} \Big(q_{(n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})} \cdot \\ \qquad\qquad \hat{\mathbf{T}}_{\langle (n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})\rangle,\langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle} \Big). \end{cases} \tag{4.6}$$

We define the chain growth function $\mathsf{growth} : \hat{S} \to [0, 1]$ such that

$$\mathsf{growth}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha.$$

The amplification ratio $\hat{\mathtt{A}}^{\bullet}_{D}$ equals the expected number of chains in L_0, i.e.,

$$\begin{aligned} \hat{\mathtt{A}}^{\bullet}_{D} &= \sum_{n_{0,1}=1}^{\infty} \cdots \Bigg(\sum_{n_{0,D}=1}^{\infty} \cdots \Bigg(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \cdot \\ &\qquad\qquad \frac{\mathsf{growth}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))}{\alpha} \Bigg) \Bigg) \\ &= \sum_{n_{0,1}=1}^{\infty} \cdots \Bigg(\sum_{n_{0,D}=1}^{\infty} \cdots \Bigg(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \cdot n_{0,D} \Bigg) \Bigg). \end{aligned}$$

Using the augmented Markov chain, we can find a lower bound of the amplification ratio as shown in Figure 23. For $D = 50$, we can find a lower bound $\hat{\mathtt{A}}^{\bullet}_{50} \geq 2.04$.

4.6.4 Achieving chain growth

Now, we show our protocol $\Pi^{\bullet}$ can achieve the chain growth property, based on the augmented Markov chain. As mentioned, we consider a hybrid experiment where all messages sent by the adversary are removed. We show that, in the hybrid experiment, the adversary cannot slow down the chain growth of the honest players. Then, we use the Chernoff bound on the augmented Markov chain to bound the chain

growth of protocol $\Pi^{\bullet}$.

Lemma 68 (Chain growth)**.** *Consider protocol* $\Pi^{\bullet}$ *in the real execution* $\mathsf{REAL}(\omega)$*. Consider an honest player* P *with the best local chain* $\mathcal{C}$ *in round* r*, and an honest player* P_1 *with the best local chain* $\mathcal{C}_1$ *in round* r_1*, where* $r_1 > r$*. Then we have* $\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t \cdot \alpha)}$*, where* $t = r_1 - r$*,* $g = (1-\delta) \cdot \alpha_0^{\bullet}$*,* $\alpha_0^{\bullet} = \hat{\mathtt{A}}_D^{\bullet} \cdot \alpha$*, and* $\delta > 0$*.*

Proof. In order to analyze the growth of the best chain from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$, we consider a hybrid execution $\mathsf{HYB}^r(\omega)$. Since the first r rounds of both the real execution $\mathsf{REAL}(\omega)$ and the hybrid execution $\mathsf{HYB}^r(\omega)$, the best chain at round r of the two executions is the same. Furthermore, based on Lemma 67, at round r_1, the best chain in the real execution $\mathsf{REAL}(\omega)$ is longer than the best chain in the hybrid execution $\mathsf{HYB}^r(\omega)$. Thus, from round r to round r_1, the chain growth in the real execution $\mathsf{REAL}(\omega)$ is greater than the chain growth in the hybrid execution $\mathsf{HYB}^r(\omega)$.

As defined in Subsection 4.5.1, let $Q = [q_s]_{s \in \hat{S}}$ be the stationary distribution over $\hat{S}$. Here, for each state $s \in \hat{S}$, the probability that the state s occurs in the random walk is $\Pr_{s' \sim Q}[s' = s] = q_s$, where the probability q_s is computed as in Equation 4.6. (Here, the state s is in the format $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$.)

We have,

$$\begin{aligned}
\mathbb{E}_{s \sim Q}\left[\mathsf{growth}(s)\right] &= \mathbb{E}_{\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle \sim Q}[\mathsf{w}(n_{0,D})] \\
&= \sum_{\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle \in \hat{S}} q_{\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle} \cdot \mathsf{w}(n_{0,D}) \\
&= \hat{\mathtt{A}}_D^{\bullet} \cdot \alpha.
\end{aligned}$$

Recall that each state in the augmented Markov chain provides a lower bound on

the number of chains in the depth-distance-based subsets. The chain growth function growth returns the probability of successfully extending a chain in the 0-depth subset L_0. (Note that from the number of chains in the depth-distance-based subsets, we can also obtain the number of chains in the depth-based subset.) Thus, the chain growth function provides a lower bound for the probability that the honest players generate a new best chain. Therefore, the augmented Markov chain $(\hat{S}, \hat{\mathbf{T}})$ and the chain growth function growth are typical to protocol $\Pi^{\bullet}$. Hence, based on Lemma 65, we have,

$$\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) < (1-\delta) \cdot \hat{\mathtt{A}}_D^{\bullet} \cdot \alpha \cdot t\right] < e^{-\Omega(t \cdot \alpha)}.$$

□

4.7 Common Prefix in Multi-Extension: A New Analysis Framework

We present a new analysis framework for examining the common prefix property in multi-extension protocols. We introduce the concepts of *virtual block-sets* and *virtual chains*. Then, we define the common prefix property w.r.t. virtual chains and prove that our protocol can achieve this property. Finally, we demonstrate that the standard common prefix property can be reduced to the common prefix w.r.t. virtual chains.

4.7.1 Virtual block-sets and virtual chains

We construct virtual block-sets and then form virtual chains based on them. Intuitively, blocks with the same height that are "close" to each other are grouped into a virtual block-set. Then, a virtual chain is formed by concatenating these virtual block-sets that are linked together. This method is intended to ensure that, at each height, honest players will only extend blocks that belong to the same virtual block-set.

We define two blocks to be "close" as follows: Given two blocks B and B' with the same height, let $\mathcal{C}$ and $\mathcal{C}'$ be the chains from the genesis block to B and B', respectively. The blocks B and B' are considered "close" to each other if the distance from $\mathcal{C}$ to $\mathcal{C}'$ is less than D, i.e. $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$.

In our protocol $\Pi^{\bullet}$, consider an honest player and let $\mathbb{C}_{\text{best}}$ be the set of the player's best chains. For any two chains $\mathcal{C}, \mathcal{C}' \in \mathbb{C}_{\text{best}}$, the distance between $\mathcal{C}$ and $\mathcal{C}'$ is less than D, i.e. $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$. Hence, if $\mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}')$, the last blocks on $\mathcal{C}$ and $\mathcal{C}'$ are "close" to each other. Note that an honest player only extends chains in the set of best chains. Thus, at each height, an honest player will only extend blocks that belong to the same virtual block-set.

Figure 32 illustrates an example of virtual block-sets with $D = 2$. In this example, the set of virtual block-sets is $\mathbb{V} = V_0, V_1, V_2, V_3, V_3', V_4, V_5$. A virtual chain is formed by linking several virtual block-sets together. A virtual block-set V is considered linked to a virtual block-set V' if there exists a block $B \in V$ and a block $B' \in V'$ such that B is linked by B'.

Definition 69 (Virtual block-sets and virtual chains)**.** *Consider an execution of protocol $\Pi^{\bullet}$, and consider an honest player with the set $\mathbb{C}$ of local chains. Let $\mathbb{B}$ be the set of all blocks on the chains in $\mathbb{C}$.*

Based on the set of block $\mathbb{B}$, we define a set $\mathbb{V}$ of **virtual block-sets***, as follows. Initially, we set $\mathbb{V} := \emptyset$. For each block $B \in \mathbb{B}$, let $\mathcal{C}$ be the chain from the genesis block to the block B. If the block B has not been added to any virtual block-set (i.e., for all $V' \in \mathbb{V}$, we have, $B \notin V'$), we build a virtual block V based on the block B as follows. Initialize that $V := \{B\}$. For any block $B' \in \mathbb{B}$, let $\mathcal{C}'$ be the chain from the genesis block to the block B'. If $\mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}')$ and $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$, we set $V := V \cup \{B'\}$. Finally, we set $\mathbb{V} := \mathbb{V} \cup \{V\}$.*

Based on the above information, we can further define a set $\mathbb{VC}$ *of* **virtual chains**, *as follows. Initialize that* $\mathbb{VC} := \{\mathcal{V}_0\}$, *where* $V_0 = \{B_0\}$ *and* B_0 *is the genesis block. For each virtual chain* $\mathcal{V} = V_0 \| V_1 \| \cdots \| V_\ell$ *(where* ℓ *is a non-negative integer) in the set* $\mathbb{VC}$, *we construct new virtual chains as follows. First, we define that* V_ℓ *is linked by the* $V_{\ell+1}$ *if there exists a block* $B_{\ell+1} \in V_{\ell+1}$ *and a block* $B_\ell \in V_\ell$ *such that* B_ℓ *is linked by* $B_{\ell+1}$[4]. *For each such virtual block-sets* $V_{\ell+1} \in \mathbb{V}$ *such that* V_ℓ *is linked by* $V_{\ell+1}$, *we construct a new virtual chain* $\mathcal{V}' := \mathcal{V} \| V$ *and set* $\mathbb{VC} := \mathbb{VC} \cup \{\mathcal{V}'\}$. *In the example in Figure 32,* $V_0 \| V_1 \| V_2 \| V_3'$ *and* $V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$ *are two virtual chains.*

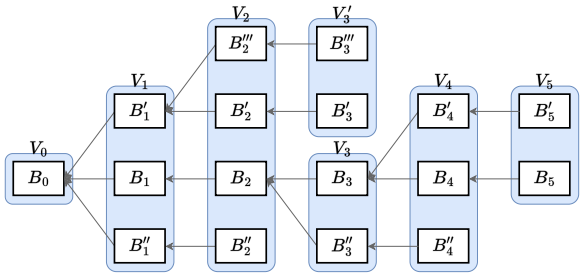


Figure 32: A toy example for the virtual block-sets and virtual chains with $D = 2$. Each block is represented by a solid rectangle and each virtual block-set is represented by a blue area that consists of multiple blocks. Here $V_0 = \{B_0\}$, $V_1 = \{B_1, B_1', B_1''\}$, $V_2 = \{B_2, B_2', B_2'', B_2'''\}$, $V_3 = \{B_3, B_3''\}$, $V_3' = \{B_3', B_3'''\}$, $V_4 = \{B_4, B_4', B_4''\}$, $V_5 = \{B_5, B_5'\}$, In this case, the best chain is $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| B_2 \| B_3 \| B_4 \| B_5$. Here, for all $i \in [0..5]$, we have $B_i \in V_i$. Thus, the best virtual chain is $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$.

[4] *In protocol* $\Pi^\bullet$, *the block* $B_{\ell+1}$ *is in the form of* $\langle \eta, r, \text{PK}, \sigma \rangle$, *where* η *is the hash value of the previous block,* r *is the current round number,* PK *is the public key of the player, and* σ *is the signature of the player over* $\langle \eta, r \rangle$. *We say* B_ℓ *is linked by* $B_{\ell+1}$ *if* $\eta = h(B_\ell)$.

We define the *best virtual chain* as the virtual chain in which each virtual block-sets in the virtual chain contains a block in the best chains. In Figure 32, the best virtual chain is $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$ since the best chain is $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| B_2 \| B_3 \| B_4 \| B_5$. We formally define the best virtual chain as follows.

Definition 70 (The best virtual chain)**.** *Let $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| \cdots \| B_\ell$ be the best chain. A virtual chain $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_\ell$ is the best virtual chain if for all $i \in [0..\ell]$, $B_i \in V_i$.*

Virtual block-set and virtual chain basics. Consider a virtual chain $\mathcal{V}$ consists of a sequence of ℓ concatenated blocks $V_0 \| V_1 \| V_2 \| \cdots \| V_\ell$, where $\ell \in \mathbb{N}$. We use $\mathcal{V}[i]$ to denote the i-th virtual block-set V_i in virtual chain $\mathcal{V}$. Here, $\imath$ denote the block height of the virtual block-set V_i in the virtual chain $\mathcal{V}$. The block height of a virtual block-set V_i is equal to the block height of all blocks in V_i. We refer to $\mathcal{V}[j, m]$, with $j \geq 0$ and $m \leq \ell$, as a sub virtual chain $V_j \| \cdots \| V_m$. If a virtual chain $\mathcal{V}$ is truncated the last κ virtual block-sets, we write $\mathcal{V}[\neg\kappa]$.

4.7.2 Unique signature scheme

In a unique signature scheme, for every possible verification key, every message to be signed, there is a unique signature. Please see Section 6.5.1 of Goldreich's textbook [53] for details. Here we include a version of the definition for syntax and properties: A unique signature scheme consists of four algorithms, a randomized key generation algorithm uKeyGen, a deterministic key verification algorithm uKeyVer, a deterministic signing algorithm uSign, and a deterministic verification algorithm uVerify; we expect for each verification key there exists only one signing key; we also expect for each pair of message and verification key, there exists only one signature. We have the following definition.

Definition 71. *We say* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a unique signature scheme, if it satisfies:*

Correctness of key generation: Honestly generated key pair can always be verified. More formally, it holds that

$$\Pr\left[(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{uKeyGen}(1^\kappa) \mid \mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}) = 1 \right] = 1.$$

Uniqueness of signing key: There does not exist two different valid signing keys for a verification key. More formally, for all PPT *adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[(\mathrm{PK}, \mathrm{SK}_1, \mathrm{SK}_2) \leftarrow \mathcal{A}(1^\kappa) \;\middle|\; \begin{array}{l} (\mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}_1) = 1) \\ \bigwedge\ (\mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}_2) = 1) \\ \bigwedge\ (\mathrm{SK}_1 \neq \mathrm{SK}_2) \end{array} \right] \leq \mathsf{negl}(\kappa).$$

Correctness of signature generation: For any message x*, it holds that*

$$\Pr\left[(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{uKeyGen}(1^\kappa); \sigma := \mathsf{uSign}(\mathrm{SK}, x) \mid (\mathsf{uVerify}(\mathrm{PK}, x, \sigma) = 1) \right] \geq 1 - \mathsf{negl}(\kappa)$$

Uniqueness of signature generation: For all PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[(\mathrm{PK}, x, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(1^\kappa) \;\middle|\; \begin{array}{l} (\mathsf{uVerify}(\mathrm{PK}, x, \sigma_1) = 1) \\ \bigwedge\ (\mathsf{uVerify}(\mathrm{PK}, x, \sigma_2) = 1) \\ \bigwedge\ (\sigma_1 \neq \sigma_2) \end{array} \right] \leq \mathsf{negl}(\kappa).$$

Unforgeability of signature generation: For all PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{uKeyGen}(1^\kappa); \\ (x, \sigma) \leftarrow \mathcal{A}^{\mathsf{uSign}(\mathrm{SK}, \cdot)}(1^\kappa) \end{array} \;\middle|\; \begin{array}{l} (\mathsf{uVerify}(\mathrm{PK}, x, \sigma) = 1) \\ \bigwedge\ ((x, \sigma) \notin Q) \end{array} \right] \leq \mathsf{negl}(\kappa),$$

where Q *is the history of queries that the adversary* $\mathcal{A}$ *made to signing oracle* $\mathsf{uSign}(\mathrm{SK}, \cdot)$*.*

Unique signature schemes and related notions have been investigated in liter-

atures (e.g., [54, 81, 76]). Please see Section 6.5.1 of Goldreich's textbook [53] for detailed discussions about the constructions. Several efficient constructions can be found in literature. For example, the well-known BLS signature [18] can be a good candidate.

4.7.3 Common prefix property w.r.t. virtual chains

We are now ready to define the common prefix property w.r.t. virtual chains. The property states that all honest players share the same common prefix of virtual chains after removing the last κ virtual blocks.

Definition 72 (Common prefix w.r.t. virtual chains)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The common prefix with respect to virtual chains, states the following: for any honest player* P' *adopting a local best virtual chain* $\mathcal{V}'$ *at round* r'*, and honest player* P *adopting a local best virtual chain* $\mathcal{V}$ *at round* r*, in the execution* $\mathsf{EXEC}_{\Pi,\mathscr{A},\mathscr{Z}}$*, where* $P', P \in \mathcal{P}$ *and* $r \leq r'$*, it holds that* $\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'$*, where* $\mathcal{V}[\neg\kappa]$ *is the virtual chain resulting from removing the last* κ *blocks.*

In order to demonstrate the common prefix property with respect to virtual chains, we introduce the concept of an *honest virtual block-set*, where the first block generated in the virtual block-set is honest. Our analysis shows that there is at most one honest virtual block-set at any block height. This means that, in order to violate the common prefix property, the adversary must extend the virtual chain as quickly as the honest players. This, in turn, requires the adversary to have control over the majority of the stake. Therefore, our protocol achieves the common prefix property under the assumption that the majority of stake is controlled by honest players.

Definition 73 (Honest virtual block-sets)**.** *Consider a virtual block-set* V*, let* B *be*

the earliest block in V, i.e., B is the block with the smallest round number[5]*. We say V is the honest if the earliest block B is generated by an honest player.*

We override the equal operator for virtual block-sets since new blocks may be added to the existing virtual block-sets through time. Intuitively, we say two virtual block-sets are equal if all the blocks in the two virtual block-sets are "close".

Definition 74 (Equal operator for virtual block-sets)**.** *Consider two virtual block-sets V_i and V_i' at the same block height i. We say V_i equals V_i' (i.e., $V_i = V_i'$) if the following constraint is satisfied: For any block $B \in V_i$ and any block $B' \in V_i'$, let $\mathcal{C}$ and $\mathcal{C}'$ be the chains from the genesis block to B and B', respectively. We have,* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$.

The equal operator for virtual block-sets is symmetric. Given that $V_i = V_i'$. For any block $B \in V_i$ and any block $B' \in V_i'$, we have, $\mathsf{distance}(\mathcal{C}' \to \mathcal{C}) = \mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$. Thus, based on Definition 74, we have, $V_i' = V_i$.

We will demonstrate that there is at most one honest virtual block-set at each block height. The definition of virtual block-sets states that honest players only extend blocks that belong to the same virtual block-set. We consider two cases as follows (see Figure 33 for an example).

- If an honest player creates a new longest chain, a new honest virtual block-set is created at the new block height, as there was no honest virtual block-set at this height previously.
- If honest players do not create a new longest chain, they can only create a new block in an existing virtual block-set.

[5] *For a block $B = \langle \eta, r, \text{PK}, \sigma \rangle$, the round number of block B is r.*

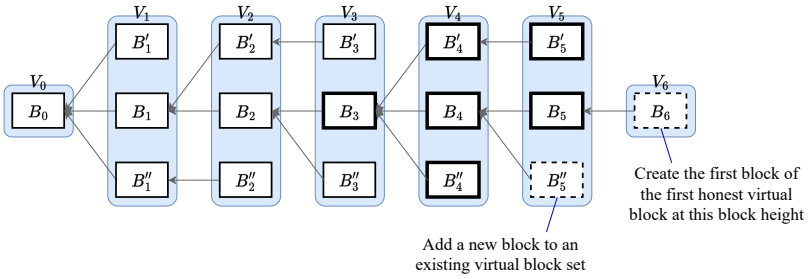


Figure 33: A toy example for illustrating an extension of honest players. Honest players extend the set of best chains from Figure 32, using 2-distance-greedy strategy. The blue blocks denote the new blocks. Here, the players generate either a new block to create a new longest chain (that is longer than the current longest chain) or a new block that is added to an existing virtual block-set.

Lemma 75. *Consider an honest player P. Let $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_\ell$ be the best virtual chain in the local state of player P at the beginning of round r, where $\ell \in \mathbb{N}$ is the length of the best chain. If player P generates a new chain $\mathcal{C} = B_0 \| B_2 \| \cdots \| B_{\ell'}$, where $\ell' \in \mathbb{N}$. Then, one of the following two conditions is true: 1) $\ell' = \ell + 1$ (a new longest chain is generated); or 2) $\ell' \leq \ell$ and $B_{\ell'} \in V_{\ell'}$ (the last block of the new chain is added to an existing virtual block-set).*

Proof. Let $\mathcal{C}_{\text{best}}$ be the best chain in the local state of player P at the beginning of round r. Let $\mathcal{C}' = \mathcal{C}[0, \ell' - 1]$. At round r, player P extend the chain $\mathcal{C}'$ by adding the block $B_{\ell'}$ to generate the new chain $\mathcal{C}$. Since the honest players only extend the chains in the set of best chains, we have, $\mathcal{C} \in \mathbb{C}_{\text{best}}$. Recall from procedure D-$\mathsf{BestChainSet}^{\bullet}$, the distance from the best chain $\mathbb{C}_{\text{best}}$ to $\mathcal{C}'$ is smaller than D, i.e., $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}') \leq D$. We consider two cases of $\mathcal{C}'$ as follows.

- The length of the chain $\mathcal{C}'$ equals ℓ, i.e., $\mathsf{len}(\mathcal{C}) = \ell$. In this case, we have,

$\ell' = \mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}'\|B) = \ell + 1$. In other words, a new best chain of length $\ell + 1$ is generated.

- The length of the chain $\mathcal{C}'$ is smaller than ℓ, i.e., $\mathsf{len}(\mathcal{C}') < \ell$. Since $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}') \leq D$, from Definition 61, we have, $\mathcal{C}_{\text{best}}[0, \ell - D] \preceq \mathcal{C}'$. Thus, $\mathcal{C}_{\text{best}}[0, \ell + 1 - D] \preceq \mathcal{C}$ (as $\mathcal{C} = \mathcal{C}'\|B$). Therefore, $\mathsf{distance}(\mathcal{C}_{\text{best}}[0, \ell'] \to \mathcal{C}) \leq D$. Plus, since $\mathcal{C}_{\text{best}}$ belongs to $\mathcal{V}_{\text{best}}$, we have, $\mathcal{C}_{\text{best}}[\ell'] \in V_{\ell'}$. Thus, based on the definition of the virtual block-set, we have, $B_{\ell'} \in V_{\ell'}$.

□

We assume $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$. If $r_1 - r$ is big enough, the adversary cannot extend the virtual chain as fast as the honest players. Given that $\Delta \cdot \alpha^\bullet \ll 1$, most of the time, there is at most one honest virtual block-set at a block height. Now, we are ready to prove the common prefix property on virtual chains.

Lemma 76 (Common prefix w.r.t. virtual chains)**.** *Assume $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$. Consider an execution of protocol $\Pi^\bullet$ with an arbitrary adversary. Consider an honest player P in round r with the local best virtual-chain $\mathcal{V}$, and an honest player P_1 in round r_1 with the local best virtual-chain $\mathcal{V}_1$, respectively, where $r_1 \geq r$. Then, we have,*

$$\Pr\left[\mathcal{V}[\neg\kappa] \preceq \mathcal{V}_1\right] \geq 1 - e^{-\Omega(\kappa)}.$$

Proof. Assuming towards a contradiction that the virtual chain $\mathcal{V}$ does not share a common prefix with the virtual chain $\mathcal{V}'$ after removing the last κ virtual block-sets, i.e., $\mathcal{V}[\neg\kappa] \not\preceq \mathcal{V}'$. Let us consider the last common virtual block-set of $\mathcal{V}$ and $\mathcal{V}'$ generated at round r_0. By the chain growth property in Lemma 68, from round r_0 to round r, the virtual chain $\mathcal{V}$ must increase in length by at least $\alpha^\bullet \cdot t$, where $t = r - r_0$. However, from Lemma 75, there can only be at most one honest virtual

block-set at any given block height. Thus, the adversary must generate at least $\alpha^{\bullet} \cdot t$ virtual block-sets from round r_0 to round r, which occurs with probability less than $e^{-\Omega(\kappa)}$. □

4.7.4 From common prefix w.r.t. virtual chains, to the standard common prefix property

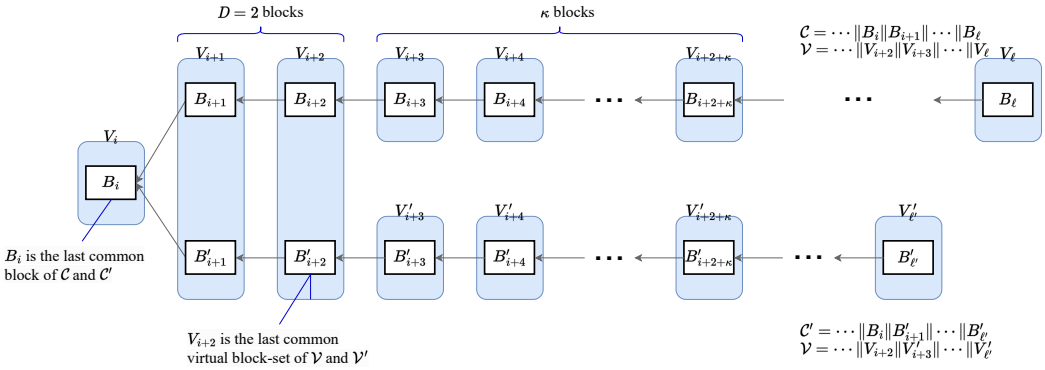


Figure 34: From common prefix w.r.t. virtual chains, to the standard common prefix property. If common prefix property does not hold, i.e., $\mathcal{C}[\neg(\kappa + D)] \preceq \mathcal{C}'$, then common prefix w.r.t. virtual chain property does not hold, i.e., $\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'$. Here, $\mathcal{C}$ belongs to $\mathcal{V}$ and $\mathcal{C}'$ belongs to $\mathcal{V}'$.

We prove the common prefix property w.r.t. virtual chains. Lemma 76 establishes that the virtual chains of any two honest players share a common prefix after removing the last κ virtual blocks. All blocks in a virtual block-set have the same common prefix after removing the last D blocks. Let V denote the last common virtual block-set between the two virtual chains. All chains in the virtual block-set V have the same common prefix after removing the last D blocks (see Figure 34). Therefore, the chains of any two honest players share the same common prefix after removing the last $\kappa + D$ blocks, thus achieving the common prefix property.

Lemma 77 (Common prefix)**.** *Assume $\alpha^{\bullet} = \lambda \cdot \beta^{\bullet}$, $\lambda > 1$. Consider an execution of protocol $\Pi^{\bullet}$ with an arbitrary adversary. Consider two honest players, P in round r with the local best chain $\mathcal{C}$, and P' in round r' with the local best chain $\mathcal{C}'$, respectively, where $r' \geq r$. Then, we have,*

$$\Pr\left[\mathcal{C}[\neg(\kappa + D)] \preceq \mathcal{C}'\right] \geq 1 - e^{-\Omega(\kappa)}.$$

Proof. Assuming toward a contradiction that $\mathcal{C}[\neg(\kappa + D)] \npreceq \mathcal{C}'$. Let $\mathcal{V}$ and $\mathcal{V}'$ be the virtual chains of $\mathcal{C}$ and $\mathcal{C}'$, respectively. Let $\ell = \mathsf{len}(\mathcal{C})$ be the length of the chain $\mathcal{C}$ and $\ell' = \ell - (\kappa + D)$. Since $\mathcal{C}[\neg(\kappa + D)] \npreceq \mathcal{C}'$, the blocks at block height ℓ' of $\mathcal{C}$ and $\mathcal{C}'$ are different, i.e., $\mathcal{C}[\ell] \neq \mathcal{C}'[\ell]$. Thus, we have, $\mathsf{distance}(\mathcal{C}[0, \ell' + D] \rightarrow \mathcal{C}'[0, \ell' + D]) > D$.

Let $\mathcal{V}[\ell'+D], \mathcal{V}'[\ell'+D]$ be the virtual block-sets at block height $\ell'+D$ of the virtual chains $\mathcal{V}$ and $\mathcal{V}'$, respectively. We have, $\mathcal{C}[\ell'+D] \in \mathcal{V}[\ell'+D]$ and $\mathcal{C}'[\ell'+D] \in \mathcal{V}'[\ell'+D]$. As $\mathsf{distance}(\mathcal{C}[0, \ell' + D] \rightarrow \mathcal{C}'[0, \ell' + D]) > D$, we have, $\mathcal{V}[\ell' + D] \neq \mathcal{V}'[\ell' + D]$. In other words, $\mathcal{V}[0, \ell' + D] \neq \mathcal{V}'[0, \ell' + D]$ or $\mathcal{V}[\neg\kappa] \npreceq \mathcal{V}'$. This contradicts the common prefix property w.r.t. virtual chains in Lemma 76.

□

4.8 Chain quality and best possible unpredictability

We show that our protocol can achieve the chain quality and the best possible unpredictability properties.

4.8.1 Chain quality

After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [88]. Intuitively, the adversary cannot extend the chain as fast as the growth rate of the best chain. Thus, some blocks on the best chain must be generated by the honest players.

Lemma 78 (Chain quality)**.** *Assume* $\alpha^\bullet = \lambda \cdot \beta^\bullet$*,* $\lambda > 1$*, and* $\delta > 0$*. Consider an execution of protocol* $\Pi^\bullet$ *with an arbitrary adversary. Consider an honest player with chain* $\mathcal{C}$*. Consider that* ℓ *consecutive blocks of* $\mathcal{C}$*, where* ℓ_{good} *blocks are generated by honest players. Then we have* $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ *where* $\mu = 1 - (1+\delta) \cdot \frac{1}{\lambda}$*.*

Proof. Assuming toward contradiction that all block from round r to round r_1 are generated by malicious players. From Lemma 68, we have, the length of the best chain from round r to round round r_1 increase by at least $(1-\delta) \cdot \alpha^\bullet \cdot t$, where $t = r_1 - r$. As all blocks from round r' to round r'' are generated by malicious players, the adversary can grow the chain with the rate $(1-\delta) \cdot \alpha^\bullet$. Recall that, the adversary can grow the chain with the rate at most $\beta^\bullet$. Thus, we have, $\beta^\bullet > (1-\delta) \cdot \alpha^\bullet$. This contradicts the assumption that $\beta^\bullet < (1-\delta) \cdot \alpha^\bullet$. □

4.8.2 Best possible unpredictability

We now show that protocol $\Pi^\bullet$ can achieve the best possible unpredictability. In our protocol, players only predict whether or not they can generate the next block, i.e., they are 2-unpredictable. In our protocol, the context of a chain is computed as the last block on the chain. Thus, the contexts of any two different chains in our protocol execution are different. In other words, the our protocol $\Pi^\bullet$ archives distinct-context-extension property. Hence, protocol $\Pi^\bullet$ achieves the best possible unpredictability.

Lemma 79. *Consider an execution of protocol* $\Pi^\bullet$ *with a set of player* $\mathcal{P}$*. For every* PPT $\mathcal{Z}, \mathcal{A}$*, for any player* $P \in \mathcal{P}$ *at any round* r*, we have,*

$$\Pr\left[\begin{array}{l|l} \mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^\bullet,\mathcal{A},\mathcal{Z}}; & \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \mathtt{VIEW}^r) & \left(\mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r', z_P^{r'}) = 0\right) \end{array}\right] > 1 - \mathsf{negl}(\kappa),$$

Proof. Assuming toward contradiction that the player P is 2-predictable at round r.

Hence, there exists a round $r' > r$ such that the adversary $\mathcal{A}$ can make an accurate prediction $z_P^{r'}$ at round r and $\mathsf{len}(\mathcal{C}^{r'}) = \mathsf{len}(\mathcal{C}^r) + 1$, where $\mathcal{C}^r$ and $\mathcal{C}^{r'}$ are the best chains at round r and r', respectively. Let (SK, PK) be the key pair of player P. In order to predict whether or not the player P can extend the chain $\mathcal{C}_P^{r'}$, the adversary must be able to compute the context η. Since the context η is computed as the hash value of the last block in the chain $\mathcal{C}^{r'}$, the adversary must know the chain $\mathcal{C}^{r'}$ to make a correct prediction on whether or not the player P can extend the chain $\mathcal{C}^{r'}$. As the chain $\mathcal{C}^{r'}$ is not generated at round r, the adversary cannot provide a accurate prediction. □

4.9 Extensions

We provide the extensions for our protocol to make it more practical. In Subsection 4.9.1, we will "upgrade" our protocol to a regular blockchain protocol so that payload (e.g., the transactions) can be included. Then, in Subsection 4.9.2, we further extend our protocol in a more realistic "non-flat" model. Finally, in Subsection 4.9.3, we follow a similar strategy in [6] to allow new players to join the system and participate in the process of extending the chains if they have their stake registered a specified number of rounds earlier.

4.9.1 Full-fledged blockchain

We extend protocol $\Pi^\bullet$ to a full blockchain protocol by using it to generate a random beacon that selects the PoS-players that can generate new main-blocks with payloads. The blocks are linked together as a hash chain called the main-chain. Each PoS-player holds a pair of keys, (SK, PK), from a unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$ and a pair of keys, $(\tilde{\text{SK}}, \tilde{\text{PK}})$, from a regular digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. We note that, to achieve adaptive security, this regular

signature scheme will be replaced by a forward-secure digital signature scheme [11].

More concretely, consider a best chain $\mathcal{C} = B_0 \| B_1 \| \cdots \| B_\ell$ with the corresponding main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1, \cdots \| \tilde{B}_\ell$. Here, the genesis block B_0 and the genesis main-block $\tilde{B}_0$ are the same. Once a new block $B_{\ell+1}$ is generated by a PoS-player, then the same PoS-player is selected to generate the new main-block $\tilde{B}_{\ell+1}$, in the following format $\tilde{B}_{\ell+1} = \langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}, \tilde{\text{PK}}, \tilde{\sigma} \rangle$ where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\text{SK}}}(\tilde{h}_\ell, B_{\ell+1}, X_{\ell+1})$, $\tilde{h}_\ell := \mathsf{hash}(\tilde{B}_\ell)$, $X_{\ell+1}$ is payload. By linking the blocks in the main blockchain to the blocks in the blockchain, the security of the main blockchain protocol can be reduced to the security of the blockchain protocol.

4.9.2 Blockchain in the non-flat model

In our previous sections, we presented our ideas in the "flat" PoS model, where all players are assumed to hold the same number of stake and have an equal chance of being selected as the winning player in each round. However, in reality, PoS players have varying amounts of stake. In this section, we will extend our design to reflect the more realistic "non-flat" model.

The genesis block, B_0, in the "non-flat" model consists of the public keys of the players, their respective stake distribution, and a randomness. Specifically, we have, $B_0 = \langle (\langle \text{PK}_1, s_1 \rangle, \langle \text{PK}_2, s_2 \rangle, \cdots, \langle \text{PK}_N, s_N \rangle), \mathsf{rand} \rangle$. The number of stakes held by each player can vary in this model. For a PoS player with the key pair (PK, SK) holding s stake at round r, the hash inequality is changed as follows:

$$\mathsf{H}(\eta, r, \text{PK}, \sigma) < s \cdot \mathsf{T}.$$

It is straightforward to see that the probability of a PoS player being selected to generate a new block is proportional to the amount of stake they control. If the player puts all their s stake in one account, their probability of being selected to sign

a PoS block is $s \cdot p$. On the other hand, if they divide their s stake into s accounts, each with one stake, the probability of an individual account being selected is p. As the outputs of the hash function are independent for different verification keys, the total probability of the player being selected is $1 - (1 - p)^s \approx s \cdot p$.

4.9.3 Defending against adaptive registration

Our design can be further improved to allow for the dynamic registration and deregistration of players during the protocol's execution. As a reminder, the chain extension process relies on the hash inequality $\mathsf{H}(\mathit{context}, \mathit{solution}) < \mathsf{T}$, where *solution* takes the form of (PK, σ). However, malicious players can use a "rejection re-sampling" strategy to generate their keys adaptively, taking advantage of the known *context*. In this strategy, a malicious player generates a key-pair $(\mathrm{PK}, \mathrm{SK})$ and then checks if the resulting (PK, σ) is a valid solution to the hash inequality. If it's not, the player repeats the key generation process. This increases the probability that the malicious player will be selected to extend the chain.

Adaptive registration. Similar to the approach in [6], we defend against rejection re-sampling attacks by requiring new players to have their stake registered for a specified number of rounds before being allowed to extend the chains. To join the protocol, player P generates two key pairs: $(\mathrm{SK}, \mathrm{PK}) \leftarrow \mathsf{uKeyGen}(1^\kappa)$ and $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. P keeps SK and $\tilde{\mathrm{SK}}$ secret and broadcasts a registration transaction. After a player has registered, they are eligible to extend the chain after η blocks have been added to the blockchain. It is important to note that players who registered prior to the start of the protocol (i.e. in the genesis block) do not have to wait η blocks before they can extend the chain.

By implementing this requirement, malicious players cannot register key pairs to extend the chains immediately. However, they can still attempt to register biased

key pairs and then extend the chains many rounds later. But since the adversary cannot accurately predict future events, they cannot choose a biased key pair that will increase their chances of extending the chain many rounds in the future.

4.10 Related Work

In this section, we summarize the existing results for the designs and analysis of PoS protocols.

4.10.1 Proof-of-stake protocols

The ideas of using coins/stake to construct cryptocurrency has been intensively considered. Since the inception of the idea in an online forum [15], several proof-of-stake proposals have been introduced or implemented (e.g., [2, 70, 99, 20, 13]). These proposals are *ad hoc* without formal security. Recently, several provably secure proof-of-stake based blockchain proposals have been developed. More details can be found below.

Bitcoin-like proof-of-stake protocols. We focus on Bitcoin-like PoS protocols; these are closely related to the results in the current writeup. All these related protocols follow the single-extension framework. These related protocols include Snow White [32], Ouroboros Praos [34] and Ouroboros Genesis [6], and a protocol by Bagaria et al. [7]. Note that, all of the above protocols are single-extension protocols (thus suffering from the impossibility result in Section 4.2).

In Snow White [32], the protocol execution is divided into epochs, where each epoch consists of $\Omega(\kappa)$ blocks (for security parameter κ). The players are selected to generate new blocks based on the public key, the current round number, and the randomness of the current epoch (via a hash inequality). The Snow White protocol is based on the Sleepy protocol [90] (in which the new players are not allowed to join

the system during the execution). The Snow White protocol allows new players to join the system but relies on external trust.

In Ouroboros Praos [34], similar to Snow White, the protocol execution is divided into epochs of $\Omega(\kappa)$ blocks. In each round, the player queries a verifiable random function (VRF) [81] to determine whether it can generate a new block; note that the input of VRF consists of the current round, the public key of the player, and the randomness of the current epoch. Here, the randomness of the epoch is computed based on the output of the VRF in the previous epoch. Note that, the protocol of Ouroboros Praos does not allow new players to join the system after the protocol execution starts. In their follow-up work, Ouroboros Genesis [6], new players are allowed to join the protocol execution securely.

In Bagaria et al. [7], similar to Praos, the players use a VRF to determine whether or not they can generate new blocks. However, here, the length of each epoch can be arbitrary. The authors also adopt the technique in [6] to allow new players to join the system. We remark that, *the work in [7] is independent and concurrent from our effort in this paper.*

BFT-like PoS protocols. Besides Bitcoin-like PoS protocols, in which the players generate blocks in a non-interactive fashion, BFT-like PoS protocols (including Algorand [23, 51], EOS [3], Dfinity [55]) have been constructed in an interactive fashion.

In Algorand [23, 51], verifiable random function (VRF) has been used for selecting a committee of players. For each player, the opportunity to be selected is proportional to the number of stake the player's account. Then, the committee members run a Byzantine Agreement (BA) sub-protocol to jointly generate a block.

EOS [3] introduces a delegated proof-of-stake protocol, in which stakeholders (those who hold the stake on the blockchain) can select block producers through a

continuous approval voting system. At the beginning of each round, 21 unique block producers are chosen based on the preference of votes cast by token holders. The selected block producers can create new blocks as long as 15 or more block producers agree.

Dfinity [55] proposes a four-layer consensus protocol to achieve consensus among players. The first layer registers the players. The second layer provides randomness for all higher layers. The third layer generates blocks. In each round, the protocol ranks the players based on the random beacon of that round. All players can generate new blocks, but each block has a different weight. The weight of the block is assigned based on the rank of the block procedure in that round. The best chain is selected as the "heaviest" chain in terms of accumulated block weight. The fourth layer provides fast finality of the block by using a threshold signature.

Note that, the above protocols (Algorand [23, 51], EOS [3], Dfinity [55]) require quadratic communication complexity to generate new blocks. HotStuff [104] uses a threshold signature scheme to achieve linear communication complexity. Specifically, the block producer, i.e., the player who generates a new block, collects votes from other players. Then, they compute and broadcast a single threshold signature that proves at least 2/3 of the players have voted for their block.

Hybrid PoS protocols using verifiable delay function (VDF). Deb et al. [36] proposed the PoSAT protocol, which is a hybrid consensus using both proof-of-stake and verifiable delay function (VDF). In the PoSAT protocol, the VDF acts as a random beacon to generate blocks. After computing a VDF, players can instantly attempt to solve a hash puzzle to check if they can extend a PoS block from the output of the VDF. Since players cannot predict the output of VDFs, the PoSAT protocol is completely unpredictable, similar to Bitcoin.

We remark that, the PoSAT protocol [36] is not a "pure" PoS protocol. In "pure" PoS protocols, the process of generating new blocks involves only the competition of "stake" (not other resources such as computing power). VDF-based PoS protocols allow the competition of sequential computation. The PoSAT protocol is based on the following assumptions: (1) both adversary and honest players have the same capability to execute sequential work: they take the same time to execute a VDF; and (2) honest players hold more stake than the adversary does. If one of the two assumptions does not hold, then the security of the PoSAT protocol cannot be ensured. However, in practice assumption (1) may not hold; it is possible that the adversary can have faster dedicated hardware for executing sequential work.

4.10.2 Security analysis for Bitcoin-like PoS protocols

Bagaria et al. [7] present a possible "balance attacks" on a multi-extension proof-of-stake protocols. We emphasize that, **their "balance attacks" cannot be launched on our protocol**. There, the adversary will try to balance the length of the two chains *by publishing the block on the shorter chain*, with the goal of maintaining two longest chains that are diverted for a long period. If the protocol is not carefully designed, the honest players may extend two chains that are diverted for a long period. Since the adversary only publishes the blocks on the shorter chain, the shorter chain will be extended faster and eventually catch up to have the same length as the other chain. Note that, in our protocol, the honest players only extend the chains that share a common prefix after removing the last few blocks. That is, the honest players will *never* extend two chains that are diverted for a long period. Thus, the adversary is not able to launch the "balance attacks" on our protocol.

Based on the analysis in [32, 34], common prefix property is guaranteed with error $e^{-\Omega(\kappa)}$ by removing the last $O(\kappa^2)$ blocks. While in Bitcoin, the consistency is

guaranteed with error $e^{-\Omega(\kappa)}$ by removing only the last $O(\kappa)$ blocks. Blum et al. [16] improve the analysis for the consistency (i.e. common prefix property) of proof-of-stake based blockchain protocols in cryptographic setting. Now, similar to Bitcoin, the consistency is guaranteed with error $e^{-\Omega(\kappa)}$ by removing only the last $O(\kappa)$ blocks. However, in [16], the "multiply honest" rounds (the rounds that have multiple honest players that can generate new blocks) are treated as "malicious" rounds (the rounds that have at least one malicious players that can generate new blocks). Kiayias et al. [64] extends the result from [16]. Here, the "multiple honest" rounds are treated as "unique honest" rounds (the rounds that have exactly one honest player that can generate a new block). Dembo et al. [38] introduces a new technique to analyze the blockchain protocols (including Bitcoin and proof-of-stake based protocols). The analysis shows that the best strategy for the adversary to break consistency is private "double-spend attack", i.e., the adversary does not contribute to the public best chain and aims to extend a private chain that is longer than the public best chain.

Unpredictability. The proof-of-stake protocols allow the players to predict whether or not they can create new blocks in the future. Indeed, in proof-of-work based protocols, the randomness is in some sense external to the blockchain. Thus, the players cannot predict whether or not they can create new blocks in the future. On the other hand, in proof-of-stake based protocols, the randomness comes from the blockchain itself. Hence, the players can predict whether or not they can create a few next block in the future. We refer to this as predictability. Brown-Cohen et al. [19] exploit the (un)predictability of proof-of-stake based protocols in a incentive-driven setting. The predictability allows the adversary to perform many incentive-driven attacks such as predictable selfish mining and predictable bribing. In this work, we investigate the unpredictability in a cryptographic setting.

4.11 Supplemental materials

4.11.1 Predictability-based attacks

We now describe the attacks where the attackers rely on the power of predictability.

Predictable selfish mining attacks. In a selfish mining attack, a player chooses to not immediately publish the blocks they have generated to the rest of the network, which undermines the fairness of the blockchain. This type of attack is more prevalent in proof-of-stake protocols, as they allow players to predict their chances of successfully mining multiple blocks in the future. Brown-Cohen et al. [19] have demonstrated a predictable selfish mining attack in proof-of-stake protocols, where players predict a specific time period in which they will generate a certain number of blocks. If the probability that other players will not generate the same number of blocks during that time period is high enough, the player can choose to keep those blocks hidden until the last block is mined. This increases the likelihood that the player's blocks will be included in the longest chain.

Predictable bribing attacks. In bribery attacks, an attacker pays players to work on specific chains in order to benefit themselves, such as supporting double spending or censorship attacks. These attacks are more dangerous in proof-of-stake protocols, as players can predict their chances of successfully mining blocks in the future. In epoch-based proof-of-stake protocols, this is particularly true at the beginning of each epoch, when an attacker can attempt to bribe players who are likely to mine new blocks. If the attacker is able to bribe enough players, they can control the majority of the blocks mined during that epoch. There are two cases to consider:

Case 1: *The confirmation time is shorter than the length of each epoch.* In this case,

the attacker can perform a double spending attack by issuing transactions at the beginning of the epoch and then hiding their blocks. At the end of the epoch, these transactions will be confirmed on the best public chain. The attacker can then publish their hidden blocks and revert the transactions they issued at the beginning.

Case 2: *The confirmation time is longer than the length of each epoch.* The attacker can perform censorship attacks by preventing certain transactions from being included on the blockchain. In each epoch, the attacker can perform a predictable bribing attack to control a majority of the blocks, which means controlling the longest chain. Since all blocks on the longest chain belong to the attacker, they can prevent any transaction from being added to the blockchain.

4.11.2 Existing single-extension proof-of-stake protocols

We now describe the existing state-of-the-art PoS protocols in [32, 34, 7] as single-extension PoS protocols.

Snow White [32]. The Snow White protocol [32] is divided into epochs, each consisting of $T_{epoch} = \Omega(\kappa)$ rounds. When players generate new blocks, they embed random seeds in those blocks. The random seeds are then used to determine which players will generate blocks in the next epoch. The four algorithms $\mathsf{Validate}$, $\mathsf{BestChain}$, $\mathsf{Context}$, and Extend are constructed as follows:

- The algorithm $\mathsf{Context}$ takes as input a chain $\mathcal{C}$ at round r and output the context η as the the context is the concatenation of the random seeds from multiple blocks in the previous epoch. Here, the function hash first truncates the blocks in the previous epoch and obtain the random seeds from those blocks. Then it concatenates all the random seeds to obtain the context. Note that,

hash can be treated as a random oracle. The random seeds in those block are random; thus the probability that two random seeds is the same is negligible.

- The algorithm Extend takes as input a context η, a round r, and a public key PK. The algorithm returns a new block if the hash value of the context, the round number, and the public key are smaller than a given threshold.

- The algorithm Validate takes as input a chain $\mathcal{C}$, a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following: 1) the context is correctly computed, 2) the hash inequality in the blocks holds, and 3) the round number of the block is smaller than the current round r.

- The algorithm BestChain takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

Ouroboros Praos [34]. The Ouroboros Praos protocol is constructed using a *Verifiable Random Function* [40] (VRF). The VRF generates a pseudorandom number with a proof of its correctness. The VRF is specified by three algorithms (Gen, Prove, Ver). The algorithm Gen takes the security parameter κ as input and outputs a key pair (SK, PK). The algorithm Prove takes the secret key SK and a message msg as input and returns a pseudorandom output σ along with a proof π. We write $(\sigma, \pi) := \mathsf{Prove}_{\text{SK}}(msg)$. The algorithm Ver takes a public key PK, a message msg, an output σ, and a proof π as input and returns 1 if the output and the proof are correct. Similar to the Snow White protocol [32], the Ouroboros Praos protocol [34] separates rounds into epochs; each epoch has $T_{epoch} = \Omega(\kappa)$ rounds. The four algorithms Validate, BestChain, Context, and Extend is constructed as follows:

- The algorithm Context takes as input a chain $\mathcal{C}$ at round r and outputs the context η as the hash value of the VRF output in the blocks in $\mathcal{C}$ that are

generated in the previous epochs. Here, the function hash can be treated as a random oracle.

- The algorithm Extend takes as input a context η, a round r, and a secret key SK. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\mathrm{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \mathtt{T}$, where $\mathtt{T}$ is the difficulty.

- The algorithm $\mathsf{Validate}$ takes as input a chain $\mathcal{C}$ and a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following conditions: 1) the context is computed correctly, 2) the VRF output in the block is computed correctly using algorithm $\mathsf{Ver}_{(\cdot)}(\cdot)$, 3) the output of the VRF is smaller than the difficulty $\mathtt{T}$, and 4) the round number in the block is smaller than the current round r.

- The algorithm $\mathsf{BestChain}$ takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

Bagaria et al. [7]. The protocol described in Bagaria et al. [7] is divided into epochs, each of which consists of $c \in \mathbb{N}$ blocks. The protocol uses a VRF to determine which players can generate new blocks. The following four algorithms are defined: $\mathsf{Validate}$, $\mathsf{BestChain}$, $\mathsf{Context}$, and Extend.

- The algorithm $\mathsf{Context}$ takes as input a chain $\mathcal{C}$ at round r and outputs the context η as the VRF output of the last block from the previous epoch. The probability that two blocks have the same VRF output equals the probability of selection two random numbers in $\{0,1\}^{\kappa}$ that are equals. As the number of blocks is polynomial in κ, the probability that there exist two blocks that have the same VRF output is negligible.

- The algorithm Extend takes as input a context η, a round r, and a secret key SK. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\mathrm{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \mathtt{T}$, where $\mathtt{T}$ is the difficulty.

- The algorithm $\mathsf{Validate}$ takes as input a chain $\mathcal{C}$ and a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following conditions: 1) the context is computed correctly, 2) the VRF output in the block is computed correctly using algorithm $\mathsf{Ver}_{(\cdot)}(\cdot)$, 3) the output of the VRF is smaller than the difficulty $\mathtt{T}$, and 4) the round number in the block is smaller than the current round r.

- The algorithm $\mathsf{BestChain}$ takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

CHAPTER 5

THE APPLICATION LAYER

In this chapter, we leverage blockchain technology to propose secure client selections protocols in federated learning process. Thanks to the immutability and transparency of blockchain, our proposed protocol enforces a random selection of clients, thereby preventing the server from manipulating the selection process.

We present the secure client selection problem in Section 5.1. In Section 5.2, we propose our protocol in a semi-malicious setting. The analysis of the protocols in the semi-malicious setting is presented in Section 5.3. Section 5.4 provides our protocol in an active setting. Experiments to evaluate our solution are given in Section 5.5. We discuss some related work in Section 5.6 and finally provide concluding remarks in Section 5.7.

5.1 Federated Learning and Secure Client Selection Problem

In this section, we first review FL protocols. Then we describe our client selection problem.

5.1.1 Federated Learning

Depending on how training data is distributed among the participants, there are two main versions of federated learning: horizontal and vertical. In this paper, we focus on horizontal FL in which different data owners hold the same set of features but different sets of samples.

Typically, an FL process follows the FedAvg framework [80] which comprises

multiple rounds. In this setting, a server and a set $\mathcal{U}$ of $n = |\mathcal{U}|$ clients participate in a collaborative learning process. Each client $u \in \mathcal{U}$ holds a training dataset D_u and agrees on a single deep learning task and model architecture to train a global model. A central server $\mathcal{S}$ keeps the parameters G^t of the global model at round t. Let x_u^t be a vector representing the parameters of the local model of client u at round t. Each training round includes the following phases:

1. *Client selection*: $\mathcal{S}$ samples a subset of m clients $\mathcal{U}' \subseteq \mathcal{U}$ and sends them the current global model G^t.

2. *Client computation*: each selected client $u \in \mathcal{U}'$ updates G^t to a new local model x_u^t by training on their private data D_u, and uploads x_u^t to the central server $\mathcal{S}$.

3. *Aggregation*: the central server $\mathcal{S}$ averages the received local models to generate a new global model as follows:

$$G^{t+1} = \frac{1}{m} \sum_{u \in \mathcal{U}'} x_u^t \tag{5.1}$$

The training continues until the global model converges.

5.1.2 Secure client selection (SCS) problem

Threat model. In our model, there are one server, n clients that execute a client selection protocol. The protocol execution comprises multiple training rounds, each consisting of multiple timesteps. In each training round, the server and clients execute a client selection protocol Π to select a set of clients. The selected clients are notified and possibly provided with proof of selection. Only selected clients will participate in the training round.

The protocol execution is directed by an environment $\mathcal{Z}(1^\kappa)$ that captures all aspects external to the protocol itself, such as inputs to the protocol (e.g., the data

of clients). Here, $\kappa \in \mathbb{N}$ is the security parameter. At the beginning of the execution, the environment $\mathcal{Z}$ initializes the server and the participating clients as either honest or corrupt. The adversary can corrupt the server and at most a fraction β of the clients.

We first consider a *semi-malicious adversary* that follows the protocol execution but can arbitrarily choose its local randomness and inputs. This semi-malicious server can be viewed as an extension of previous work on secure aggregation in FL [17, 106, 5]. In Section 5.4, we will also consider an *active adversary* that can divert from the protocol.

Problem definition. We define a new problem, called *secure client selection (SCS)* problem that asks for a protocol Π, executed by a server $\mathcal{S}$ and a set of clients $\mathcal{U}$, to select a subset of clients in each training round in FL. At the end of the execution of the protocol Π, for each client j, the server $\mathcal{S}$ sends a collections of proofs $\{\omega_j^{(i)}\}_{i\in\mathcal{U}}$, in which $\omega_j^{(i)}$ is either empty or *a proof on whether or not the client i is selected*. Importantly, the designed protocol needs to satisfy three security properties, namely, *pool consistency, pool quality, and anti-targeting*.

Definition 80 (Secure client selection problem)**.** *Let st_j be the local state of the client $j \in \mathcal{U}$ at the end of a training round. We say Π is secure iff there exists a predicate PVer_Π that takes the state st_j, a proof $\omega_j^{(i)}$ (provided by server $\mathcal{S}$) as input and outputs*

$$\mathsf{PVer}_\Pi(\mathsf{st}_j, \omega_j^{(i)}) = \begin{cases} 1 & \text{if } i \text{ is selected in the view of } j, \\ 0 & \text{if } i \text{ is not selected in the view of } j, \\ \perp & \text{if } \omega_j^{(i)} \text{ is empty or invalid,} \end{cases}$$

with the following properties:

- Pool consistency*: Let $\mathcal{H}$ be the set of honest clients, $\forall i \in \mathcal{U}$ and $\forall j_1, j_2 \in \mathcal{H}$,*

$$\Pr\left[\exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} \;\middle|\; \begin{array}{l} \mathit{PVer}_{\Pi}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ \mathit{PVer}_{\Pi}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array}\right] \leq e^{-\Omega(\kappa)},$$

where κ is the security parameter.

- γ-pool quality for $\gamma \in (0,1)$*: Let $\mathcal{P}$ be the set of selected clients, defined as:*

$$\mathcal{P} = \{i \in U : \exists j \in \mathcal{H}\ s.t.\ \mathit{PVer}_{\Pi}(\mathsf{st}_j, \omega_j^{(i)}) = 1\}.$$

We have:

$$\Pr\left[\frac{\mathcal{H} \cap \mathcal{P}}{\mathcal{P}} \geq \gamma\right] \geq 1 - e^{-\Omega(\kappa)}.$$

- Anti-targeting*: Let $c = \frac{m}{n}$ be the* selection probability*,*

$$|\Pr[i \in \mathcal{P}] - c| \leq e^{-\Omega(\kappa)},\ \forall i \in \mathcal{U}.$$

5.2 Defending against semi-malicious adversaries

We present our defense against client selection attacks in the semi-malicious setting. We begin with identifying three essential security properties for a secure client selection in Subsection 5.1.2. Then, in Subsection 5.2.1, we propose our blockchain-based protocol (SeP) that achieves those properties. Finally, we extend SeP into CoSeP protocol to reduce significantly the communication complexity.

5.2.1 Secure protocol (SeP) for SCS problem

We propose a secure protocol (SeP) for SCS problem that use blockchain as 1) a source of public randomness, and 2) a distributed database to store the information of the selected clients. Blockchain, introduced in [83], is a type of distributed ledger, jointly maintained by a set of nodes in a network, called miners. Blockchain can

provide guarantees on the correctness (i.e. tamper-resistance) and security of the ledger without the need of trust on a central trusted party. In this work, we consider the Bitcoin protocol, in which miners compete to solve a PoW puzzle.

We remark that using blockchain to solve the secure client selection problem may introduce a different attack vector. It is possible for an adversary with sufficient mining power to revert the last few blocks in the blockchain [46]. This can lead to (temporary) inconsistent views among the participants, compromising the security of the protocol.

The protocol SeP consists of a one-time registration phase and multiple training rounds. In the registration phase, the clients submit their public keys to the blockchain. In our protocols, the client selection will happen over a sufficiently large span of blocks to ensure that all the participants, including the clients and the server, will arrive to the same view on the set of selected clients.

Algorithm 10: Protocol SeP.

Registration phase : All clients submit their public keys to the blockchain.
One trainning round: Consider a training round that starts at block height (BH) ℓ.

Step 1 (BH ℓ): *Randomness extraction.*
The server and clients extract the randomness rnd from the blockchain.
Step 2 (BH $\ell + \tau$): *Random selection.*
Each client i proceeds as follows.
    If $H(\mathsf{rnd}, \mathsf{pk}_i) < c2^{\kappa}$ **then**
        Submit a selection transaction that consists of pk_i to the blockchain

Each training round of the protocol SeP consists of 2 steps: (1) randomness extraction and (2) random selection. In step 1 (randomness extraction), all clients and the server compute locally a random token rnd by hashing together the block headers in the previous round. In step 2, the clients use a hash function to determine whether or not they get selected. Each selected client will submit a transaction to the

blockchain to announce that she/he is selected. We provide the details for the steps in the client selection protocol in Algorithm 10. We use the height of the blockchain, or *block height* (BH), to measure time. We select a parameter $\tau = \Omega(\kappa)$ so that sent messages are received and submitted transactions are finalized within 2τ blocks. For a training round started at BH ℓ, the client selection protocol is executed between block heights ℓ and $\ell + 2\tau$.

Step 1: Randomness extraction. We follow the scheme to extract the randomness in [35]. At block height ℓ, the server and all clients compute a randomness rnd by hashing together the block headers of the first τ blocks created during the previous training round. The chain quality of the blockchain means that, with high probability, at least one of those blocks must be from an honest miner [46]. Thus, rnd includes at least one unbiased random source.

Step 2: Random selection. After extracting the randomness, each client i uses the hash function to check whether or not she/he is selected in this round. Here, a hash function H, takes as input an arbitrary-length string in $\{0,1\}^*$ and output a random string in $\{0,1\}^\kappa$.

We say a client i is qualified if $H(\mathsf{rnd}, \mathsf{pk}_i) < c2^\kappa$. Here, $c = \frac{m}{n}$ is the *selection probability*, i.e., the fraction of selected clients per round. If the client i is qualified, she/he submits a transaction that consists of the public key pk_i to the blockchain. After the transaction is included in the blockchain, the client i is selected.

Proof of membership. For a client i, the proof of membership $\omega_i^{(i)}$ includes the selection transaction that consists of the public key pk_i.

Pool membership verification function. For each client j with the state st_j, the function $\mathsf{PVer}(\mathsf{st}_j, \omega_j^{(i)})$ extracts the blockchain C_j from the local state st_j and then proceeds as follows. First, the function verifies whether or not the selection transaction is included in the header blockchain C_j. If the condition does not hold, it returns

$\perp$. If the condition holds, i.e., the proof $\omega_j^{(i)}$ is valid, the function extracts the randomness rnd from the blockchain. Then, the function verifies $H(\mathsf{rnd}, \mathsf{pk}_i) < c2^\kappa$. If the condition holds, it returns 1, i.e., the client i is selected. Otherwise, the function returns 0, i.e., the client i is not selected.

While the protocol SeP is secure, it requires a huge amount of communication for both server and clients. In protocol SeP, both server and clients must download and verify all transactions on the blockchain. Plus, the direct communication between the clients and the blockchain is not efficient. In the registration phase, each client submits a transaction to register its public key. In each training round, each selected client needs to submit a transaction. This costs a huge amount of communication.

5.2.2 Communication-efficient and secure protocol (CoSeP) for SCS problem

We present a communication-efficient and secure protocol (CoSeP) using *light-weight blockchain clients* and *pass-through communication* to optimize the communication complexity.

Light-weight blockchain clients. In a blockchain system, there are light-weight SPV (Simplified Payment Verification) nodes that only need to download the block headers. A SPV node can verify that a transaction is included in the Bitcoin blockchain by requesting a proof from a full node. In protocol CoSeP, the clients only need to run the SPV nodes.

Pass-through communication. Instead of using direct communication between the clients, the server will gather the data from the clients and only submit a small commitment of those data to the blockchain. The server can provide a proof of that some data from the client was used to compute the commitment. In our protocol, we using the sorted Merkle tree [31] to implement the commitment.

Membership proof with sorted Merkle tree. Given a set of l values $X = \{d_1, d_2, \ldots, d_l\}$, a Merkle tree is a binary tree constructed over the hash values of d_i. The root of the Merkle tree, denoted by $\mathsf{MRoot}(X)$, can be used as a succinct representation of all the values. Knowing $\mathsf{MRoot}(X)$, we can construct a membership/non-membership proof of size $O(\log l)$ to prove whether a value x appears in X. Such a proof, denoted by $\mathsf{MProof}(x \stackrel{?}{\in} X)$, can also be verified in a time $O(\log l)$.

Verifiable random functions (VRF). To enhance the privacy of the selected clients, instead of using hash function, we use a cryptographic tool called *verifiable random function* [41]. By using the VRF, the adversary cannot determine whether or not a client is qualified.

VRF is a public-key pseudorandom function that provides proofs showing that its outputs were calculated correctly and randomly, i.e., hard to predict. Consider a user with secret and public keys sk and pk. The user can use VRF to generate a function output σ and a proof π for any input value x by running a function $\mathsf{VRFprove}_{\mathsf{sk}}(x)$. Everyone else, using the proof π and the public key pk, can check that the output σ was calculated correctly by calling a function $\mathsf{VRFverify}(\mathsf{pk}, \sigma, \pi)$. Yet, the proof π and the output σ does not reveal any information on the secret key sk.

In the registration phase, clients registered their public keys with the server. Let $\mathcal{U}$ be the set of registered clients, the server submits a registration transaction containing the Merkle root $\mathsf{MRoot}(\mathcal{U})$ to blockchain and sends $\mathsf{MProof}(\mathsf{pk}_i \stackrel{?}{\in} \mathcal{U})$ to each client i.

At the beginning of each training round, a random subset of clients will be selected based on a public randomness that is obtained from a blockchain. As shown in Figure 35, the selection protocol consists of 2 steps: (1) randomness extraction and (2) VRF-based random election. Step 1 is the same as in Algorithm 10. In step 2, verifiable random functions (VRFs) [41], taking the client's public key and rnd as

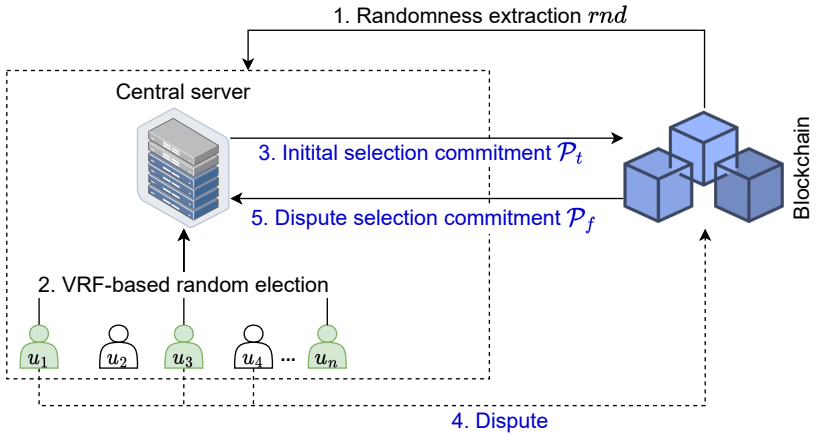


Figure 35: The steps of efficient client selection protocols in each training round. The protocol CoSeP in a semi-malicious setting consists of two steps: (1) Randomness extraction and (2) VRF-based random election. In the active setting in Section 5.4, CoSeP will be extended into a new protocol CoSeP$_+$ with 3 additional steps: (3) Initial selection commitment, (4) Dispute, and (5) Dispute selection commitment.

inputs, are employed to determine which clients are selected. We provide the details for the steps in protocol CoSeP in Algorithm 11. For a training round started at BH ℓ, the client selection protocol is executed between block heights 2ℓ and $\ell + 2\tau$.

Step 2: VRF-based random election. After extracting the randomness, each client i uses the VRF to check whether or not she/he is selected in this round. The client i computes the output σ_i and the proof π_i of the VRF based on the randomness rnd, i.e., $(\sigma_i, \pi_i) \leftarrow \mathsf{VRFprove}_{\mathsf{sk}_i}(\mathsf{rnd})$. If the VRF output σ_i is smaller than a given threshold, i.e., $\sigma_i < c2^{\kappa}$, the client i is *qualified* to be selected. Here, $c = \frac{m}{n}$ is the *selection probability*, i.e., the fraction of selected clients per round. If the client i is qualified, she/he sends a message $(\sigma_i, \pi_i, \mathsf{pk}_i)$ to the server. Since both honest and colluding clients follows the protocol, all qualified clients will send the proofs to the server.

Algorithm 11: Communication-efficient protocol CoSeP for semi-malicious settings

Registration phase : The server submits a registration transaction containing the Merkle root that is computed from the clients' public keys.

One trainning round: Consider a training round that starts at block height (BH) ℓ.

Step 1 is the same as in Algorithm 10.

Step 2 (BH ℓ): *VRF-based random election.*

Each client i proceeds as follows.

  Computes $(\sigma_i, \pi_i) \leftarrow \mathsf{VRFprove}_{\mathsf{sk}_i}(\mathsf{rnd})$

  If $\sigma_i < c2^\kappa$ (i.e., the client i is qualified) **then**

    Sends the proof $(\sigma_i, \pi_i, \mathsf{pk}_i)$ to the server.

Let $\mathcal{P}_t$ be the set of public keys of qualified clients that are verified by the server. The server computes the Merkle tree root $\mathsf{MRoot}(\mathcal{P}_t)$ and send the Merkle proof $\mathsf{MProof}(\mathsf{pk}_i \stackrel{?}{\in} \mathcal{P}_t)$ for each client $i \in \mathcal{U}$.

Proof of membership. For each qualified client i, the server provides the proof of membership $\omega_i^{(i)}$ of a client i. The proof consists of (1) the VRF output, proof, and the public key $(\sigma_i, \pi_i, \mathsf{pk}_i)$ of the client i , (2) the Merkle root MRoot_t, and (3) the Merkle proof $\mathsf{MProof}(\mathsf{pk}_i \stackrel{?}{\in} \mathcal{P}_t)$.

Pool membership verification function. For each client j with the state st_j, the function $\mathsf{PVer}(\mathsf{st}_j, \omega_j^{(i)})$ extracts the blockchain C_j from the local state st_j and then proceeds as follows. The function extract the randomness rnd as in step 1 and verifies $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i, \pi_i) = 1$. If the condition does not hold, it returns $\perp$. If all conditions hold, i.e., the proof $\omega_j^{(i)}$ is valid, the function verifies (1) $\sigma_i < c2^\kappa$, and (2) pk_i is included in MRoot_t. If those conditions hold, it returns 1, the client i is selected, and 0, otherwise.

5.3 Security Analysis

We summarize the security properties of blockchain in [46] in Subsection 5.3.1 to provide the foundation for the security analysis of SeP and CoSeP in Subsections 5.3.2 and 5.3.3, respectively. Finally, we summarize the communication complexity of each protocol in Subsection 5.3.4.

5.3.1 Security of Bitcoin protocol in [46]

We extend our model in Section 5.1.2 to include the miners, who execute the blockchain protocol. In each round, each miner makes attempts to generate a new block by solving a hash inequality. Let $H(\cdot)$ be a hash function that takes as input an arbitrary length string in $\{0,1\}^*$ and outputs a string of length κ (where κ is security parameter) in $\{0,1\}^\kappa$. A miner can generate a new block if it can find a random *nonce* such that the hash value $H(\cdot, nonce)$ is smaller than a given threshold. The adversary can corrupt to up a fraction ρ of the miner.

Based on the analysis in [46], under the honest majority assumption ($\rho < 1/2$), the Bitcoin protocol achieves 1) *persistence*: all honest miners have the same view of the ledger; and 2) *liveness*: the valid transactions will eventually be added to the ledger. The persistence and liveness properties are proved based on three properties, namely chain growth, common prefix, and chain quality.

5.3.2 Security analysis of protocol SeP

Pool consistency. In protocol SeP, a client is selected if 1) the hash value of the randomness and its public key is smaller than a given threshold; and 2) a *selection transaction* that consists of its public key is included in the blockchain. We can prove the pool consistency by showing that all honest clients have the same view on 1) the

randomness and 2) the list of selection transactions.

First, we show that all honest clients obtain the same randomness. Indeed, in protocol SeP, the randomness is computed based on the hash values of the first τ blocks in the previous training round. As each training round consists of 2τ blocks, based on the common prefix property, we can ensure that all honest clients have the same view on the first τ blocks in the previous training round. Hence, all honest clients obtain the same value of randomness.

Secondly, we prove that all honest clients have the same view of the list of selection transactions. In protocol SeP, all selection transactions are included before block height $\ell + \tau$. At the end of the client selection protocol (block height $\ell + 2\tau$), all selection transactions are included in the ledgers of the honest clients. Based on persistency property, we can guarantee that all honest clients have the same view on those selection transactions.

Lemma 81 (Pool consistency)**.** *For any client $i \in \mathcal{U}$, and any honest clients j_1, j_2, we have,*

$$\Pr\left[\begin{array}{l|l} \exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} & \mathit{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ & \mathit{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array}\right] \leq e^{-\Omega(\kappa)}.$$

Proof. Consider two honest clients j_1, j_2, let $\mathsf{I}(j_1, j_2)$ be the event where the views of j_1 and j_2 on the set of selected clients are different, i.e.,

$$\Pr[\mathsf{I}(j_1, j_2)] = \Pr\left[\begin{array}{l|l} \exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} & \mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ & \mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array}\right].$$

Assume toward contradiction that there exists a non-negligible number p such that

$$\Pr[\mathsf{I}(j_1, j_2)] > p.$$

Let S be the event where the blockchain protocol is secure, i.e., it achieves chain growth, chain quality, common prefix, persistency, and liveness properties. Given that the blockchain protocol is secure, we will show that both j_1 and j_2 obtain the same randomness rnd. Indeed, as the blockchain protocol is secure, all honest clients obtain the same prefix of the chains after removing the last τ blocks. In step 1 of the client selection protocol, the randomness rnd is computed as the hash value of the first τ blocks in the previous training round. As each training round last more than 2τ blocks, all honest clients will have the same view on the first τ blocks of the previous training round. Hence, all clients obtain the same randomness rnd.

Given the event when the blockchain protocol is secure, we consider the case such that there exists $\omega_{j_1}^{(i)}, \omega_{j_2}^{(i)}$ such that $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1$ and $\mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0$. In this case, the clients j_1 and j_2 must obtain two different selection transactions that consist of two different public key of the client i. This contradict the assumption that the public keys of the clients are known by every nodes. □

Pool quality. We prove that the fraction of honest selected clients is proportional to the fraction of honest clients. As the output of the hash function is random, we can guarantee the qualified clients are uniformly selected from the set of clients. Plus, the semi-malicious adversary strictly follows the protocol. Hence, all the qualified clients will be selected.

For a client $i \in \mathcal{U}$, let x_i is the Bernoulli random variable that represents where or not the client i is qualified, i.e.,

$$x_i = \begin{cases} 1 \text{ if } i \text{ is qualified,} \\ 0 \text{ otherwise.} \end{cases}$$

As the output of the hash function is random, we have,

$$\Pr[x_i = 1] = c.$$

Lemma 82 (Pool quality)**.** *Let $\mathcal{H}$ be the set of honest clients in the set of selected clients $\mathcal{P}$. For $\epsilon > 0$, we have,*

$$\Pr[\frac{\mathcal{H} \cap \mathcal{P}}{\mathcal{P}} \geq \alpha(1-\epsilon)] \geq 1 - e^{-\Omega(nc - \log \kappa)} - e^{-\Omega(\kappa)},$$

where n is the number of clients, $\alpha = 1 - \beta$ is the fraction of honest clients, and c is the selection probability.

Proof. We prove by bounding the number of honest qualified clients. Then, we will show that all qualified clients will be selected, since the semi-malicious server and clients still follow the protocol.

Let $\mathcal{P}'$ be the set of qualified clients, i.e., the clients having VRF outputs smaller than $c2^\kappa$. Let $\mathcal{H}'$ and $\mathcal{M}'$ be the set of honest and colluding clients in $\mathcal{P}'$, respectively.

Using the Chernoff bound on the set of $n\beta$ colluding clients $\mathcal{U}'$, for any $\epsilon' > 0$, we have,

$$\Pr[\sum_{i \in \mathcal{U}'} x_i \geq (1+\epsilon')n\beta c] \leq e^{-\Omega(nc)},$$
$$\Rightarrow \Pr[|\mathcal{M}'| \geq (1+\epsilon')n\beta c] \leq e^{-\Omega(nc)}.$$

Using the Chernoff bound on the set of $n\alpha$ honest clients $\mathcal{U} \setminus \mathcal{U}'$, for any $\epsilon' > 0$, we have,

$$\Pr[\sum_{i \in \mathcal{U} \setminus \mathcal{U}'} x_i \leq (1-\epsilon')n\alpha c] \leq e^{-\Omega(nc)},$$
$$\Rightarrow \Pr[|\mathcal{H}'| \leq (1-\epsilon')n\alpha c] \leq e^{-\Omega(nc)}.$$

By choosing ϵ such that $\epsilon = 1 - \frac{1-\epsilon'}{1+\epsilon'}$, we have,

$$\Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{M}'|} \leq \frac{\alpha}{\beta}(1-\epsilon)\right] \leq e^{-\Omega(nc)}.$$

Thus, we have,

$$\begin{aligned}\Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{P}'|} \leq \alpha(1-\epsilon)\right] &= \Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{M}'| + |\mathcal{H}'|} \leq \alpha(1-\epsilon)\right] \\ &\leq \Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{M}'|} \leq \frac{\alpha}{\beta}(1-\epsilon)\right] \leq e^{-\Omega(nc)}.\end{aligned}$$

As the clients (both honest and colluding) follows the protocols, if they are qualified, they will send the VRF proofs to the server (step 2 in Algorithm 10). Since the semi-malicious server still follow the protocol, all qualified clients will be selected. We have,

$$\Pr[\mathcal{P} = \mathcal{P}'] \geq \Pr[\mathsf{S}] \geq 1 - e^{-\Omega(\kappa)}.$$

In other words, we have, $\Pr[\mathcal{P} \neq \mathcal{P}'] < e^{-\Omega(\kappa)}$. Therefore, we have,

$$\begin{aligned}&\Pr\left[\frac{|\mathcal{H}|}{|\mathcal{P}|} \geq \alpha(1-\epsilon)\right] \geq \Pr\left[\frac{|\mathcal{H}|}{|\mathcal{P}|} \geq \alpha(1-\epsilon) \wedge \mathsf{S}\right] \\ =&\Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{P}'|} \geq \alpha(1-\epsilon) \wedge \mathsf{S}\right] \geq \Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{P}'|} \geq \alpha(1-\epsilon)\right] - (1 - \Pr[\mathsf{S}]) \\ \geq&1 - e^{-\Omega(nc)} - e^{-\Omega(\kappa)}.\end{aligned}$$

□

Anti-targeting. Since all qualified clients are selected, the probability that an honest client is selected is the same as the probability that an honest client is qualified. We can prove the anti-targeting property as follows.

Lemma 83 (Anti-targeting)**.** *For any honest client i,*

$$|\Pr[i \in \mathcal{P}] - c| = e^{-\Omega(\kappa)},$$

where c is the selection probability.

Proof. As we have shown in the proof of Lemma 87, if the blockchain protocol is secure, all qualified clients are selected. Thus, we have,

$$\begin{aligned}\Pr[i \in \mathcal{P}] &\geq \Pr[i \in \mathcal{P} \wedge \mathsf{S}] = \Pr[x_i = 1 \wedge \mathsf{S}] \\ &\geq \Pr[x_i = 1] - (1 - \Pr[\mathsf{S}]) \geq c - e^{-\Omega(\kappa)}.\end{aligned}$$

Plus, a selected clients must be qualified. Thus, we have,

$$\Pr[i \in \mathcal{P}] \leq \Pr[x_i = 1] \leq c + e^{-\Omega(\kappa)}.$$

□

Together, Lemmas 81, 82, and 83 yield the security proof of the protocol SeP in semi-malicious settings.

Theorem 84. *The client selection protocol SeP achieves pool quality, pool consistency, and anti-targeting properties in the presence of a semi-malicious adversary.*

5.3.3 Security analysis of protocol CoSeP

Pool consistency. In protocol CoSeP, a client is selected if 1) it is qualified, i.e., its VRF output on the randomness is smaller than the given threshold and 2) its public key is included in the Merkle tree root, which is computed by the server. Similar to the analysis in Subsection 5.3.2, we can show that all honest clients obtain the same value of the randomness. Plus, as the semi-malicious server strictly follows the protocol, it shares the same Merkle tree to all clients. Hence, we can guarantee the pool consistency of protocol CoSeP.

Lemma 85 (Pool consistency)**.** *For any client $i \in \mathcal{U}$, and any honest clients j_1, j_2, we have,*

$$\Pr\left[\begin{array}{l|l} \exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} & \begin{array}{l} \mathit{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ \mathit{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array} \end{array}\right] \leq e^{-\Omega(\kappa)}.$$

Proof. Consider two honest clients j_1, j_2, let $\mathsf{I}(j_1, j_2)$ be the event where the views of j_1 and j_2 on the set of selected clients are different, i.e.,

$$\Pr[\mathsf{I}(j_1, j_2)] = \Pr\left[\begin{array}{l|l} \exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} & \begin{array}{l} \mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ \mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array} \end{array}\right].$$

Assume toward contradiction that there exists a non-negligible number p such that

$$\Pr[\mathsf{I}(j_1, j_2)] > p.$$

Similar to Lemma 81, given the event S, i.e., the blockchain protocol is secure, we have both j_1 and j_2 obtain the same randomness rnd.

We obtains (σ_i, π_i) from $\omega_{j_1}^{(i)}$ and (σ_i', π_i') from $\omega_{j_2}^{(i)}$. As $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1$, we have, $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i, \pi_i) = 1$ and $\sigma_i < c2^\kappa$. As $\mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0$, we have, $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i', \pi_i') = 1$ and $\sigma_i' > c2^\kappa$. Thus, we have $\sigma_i \neq \sigma_i'$. This contradict the uniqueness property of the VRF.

□

Pool quality. We prove that the fraction of honest selected clients is proportional to the fraction of honest clients. Here, the VRFs guarantee that the qualified clients are uniformly selected. Then, we can follow the same proofs as in Subsection 5.3.2 to prove the pool quality of protocol CoSeP.

We first show that the VRFs guarantee the randomness in selecting the qualified clients. This is given since the VRF output is pseudo random.

Lemma 86. *For any client $i \in \mathcal{U}$, we have*

$$|\Pr[x_i = 1] - c| < e^{\Omega(\kappa)}. \tag{5.2}$$

Proof. First, we prove that $\Pr[x_i = 1] > c - e^{\Omega(\kappa)}$. Assume toward contradiction that there exists there exists a non-negligible number p such that

$$\Pr[x_i = 1] < c - p.$$

Let sk_i be the private key for VRF of the client i and $(\sigma_i, \pi_i) := \mathsf{VRFprove}_{\mathsf{sk}_i}(\mathsf{rnd})$. We have,

$$\Pr\left[\sigma_i < c2^\kappa\right] < c - p$$

We will construct an adversary $\mathcal{A}$ that is given query to the oracle $\mathsf{VRFprove}_{\mathsf{sk}_i}$ as follows.

- Set $x = \mathsf{rnd}$ and send x to the prover.
- Upon receiving y_b from the prover.
- If $y_b < c2^\kappa$, return $b' = 1$. Otherwise, return $b' = 0$.

We have,

$$\begin{aligned}
\Pr[b = b'] &= \frac{1}{2}\left(\Pr\left[y_0 \geq c2^\kappa\right] + \Pr\left[y_1 < c2^\kappa\right]\right) \\
&= \frac{1}{2}\left(\Pr\left[\sigma_i \geq c2^\kappa\right] + c\right) \\
&> \frac{1}{2}\left(1 - c + p + c\right) = \frac{1}{2}\left(1 + p\right).
\end{aligned}$$

This contradicts the pseudorandomness property of VRF.

Similarly, we can prove that $\Pr[x_i = 1] < c + e^{\Omega(\kappa)}$. Thus, we can conclude that

$$|\Pr[x_i = 1] - c| < e^{\Omega(\kappa)}.$$

□

Follow the same proof as in Lemma 82, we can show the pool quality of protocol CoSeP.

Lemma 87 (Pool quality)**.** *Let $\mathcal{H}$ be the set of honest clients in the set of selected clients $\mathcal{P}$. For $\epsilon > 0$, we have,*

$$\Pr[\frac{\mathcal{H} \cap \mathcal{P}}{\mathcal{P}} \geq \alpha(1-\epsilon)] \geq 1 - e^{-\Omega(nc)} - e^{-\Omega(\kappa)},$$

where n is the number of clients, $\alpha = 1 - \beta$ is the fraction of honest clients, and c is the selection probability.

Anti-targeting. Based on Lemma 86, we can follow the same proof as in Lemma 95 to show the anti-targeting property of protocol CoSeP.

Lemma 88 (Anti-targeting)**.** *For any honest client i,*

$$|\Pr[i \in \mathcal{P}] - c| \leq e^{-\Omega(\kappa)},$$

where c is the selection probability.

Together, lemmas 85, 87, 88 yield the security proof CoSeP of the protocol in semi-malicious settings.

Theorem 89. *The client selection protocol achieves pool quality, pool consistency, and anti-targeting properties in the presence of a semi-malicious adversary.*

5.3.4 Communication complexity

We summarize the communication complexity of protocols SeP, CoSeP, and CoSeP$_+$ in Table 3.

Protocols	Semi-malicious setting		Active setting
	SeP	CoSeP	CoSeP$_+$
Client	$O(\tau b + nc)$	$O(\tau)$	$O(\tau)$
Server	$O(\tau b + nc)$	$O(\tau b + nc)$	$O(\tau b + nc(1+d))$
Miner	$O(\tau b + nc)$	$O(\tau b)$	$O(\tau b + ncd)$

Table 3.: Summary of the communication complexity of 3 protocols. Here, n is the number of clients, c is the probability that a client is selected in each training round, $\tau = \Omega(\kappa)$ (where κ is the security parameter) is the length (in block height) of each training round, b is the size of each block, and d is the fraction of the dispute clients.

Let b be the average size of a block[1] without including the transactions of the client selection protocol. We have the follow lemmas.

Lemma 90 (Communication complexity of protocol SeP)**.** *In each training round of the protocol SeP, the communication complexity of each client, the server, each miner are $O(\tau b + nc)$.*

Proof. In each training round of the protocol SeP, all qualified clients will submit a transaction to the blockchain. Hence, the average transaction for each training round is nc. As each training round last for 2τ blocks, the total size of the blockchain for each training round is $2\tau b + nc$. As everyone need to download the blockchain, the total communication complexity of each node is $2\tau b + nc$. □

[1]If the blockchain is only used for the client selection protocol, we have $b = O(1)$.

Lemma 91 (Communication complexity of protocol CoSeP)**.** *In each training round of the protocol CoSeP, the communication complexity of each client, the server, each miner are $O(\tau)$, $O(\tau b + nc)$, and $O(\tau b)$, respectively.*

Proof. In the protocol CoSeP, the server and the clients do not submit any transaction to the blockchain. Thus, the total size of the blockchain for each training round is $2\tau b$. The communication complexity of each miner is $2\tau b$. The server also receives the VRF proof from nc qualified clients. Hence, the communication complexity of the server is $2\tau b + nc$. For the clients, they only need to maintain the header of blocks. As the size of each header is $O(1)$, the communication complexity of each client is 2τ. □

Let $d \in (0, 1)$ be the fraction of dispute clients. We show the communication complexity of protocol CoSeP$_+$ as follows.

Lemma 92 (Communication complexity of protocol CoSeP$_+$)**.** *In each training round of the protocol CoSeP$_+$, the communication complexity of each client, the server, each miner are $O(\tau)$, $O(\tau b + nc(1 + d))$, and $O(\tau b + ncd)$, respectively.*

Proof. In the protocol CoSeP, the server submit 2 transactions and each dispute client submit one transaction to the blockchain. Thus, the total size of the blockchain for each training round is $3\tau b + ncd$. The communication complexity of each miner is $3\tau b + ncd$. The server also receives the VRF proof from nc qualified clients. Hence, the communication complexity of the server is $3\tau b + nc(d + 1)$. For the clients, they only need to maintain the header of blocks. As the size of each header is $O(1)$, the communication complexity of each client is 3τ. □

5.4 Defending against active adversaries

We now consider an active setting where the adversary can divert from the protocol. We extend the protocol CoSeP to design a new protocol, called CoSeP$_+$ (Subsection 5.4.1), in active setting . Then, in Subsection 5.4.2, we provide the security analysis of the protocol CoSeP$_+$.

5.4.1 Protocol CoSeP$_+$

Algorithm 12: Protocol CoSeP$_+$ in active settings with three additional steps

Registration phase, steps 1 and 2, and block height settings are the same as those in Algorithm 11.
Step 3 (BH $\ell + \tau$): *Initial selection commitment.*
The server proceeds as follows.
    Upon receiving a proof $(\sigma_i, \pi_i, \mathsf{pk}_i)$ from client i
        If $\mathsf{Verify}_{\mathsf{pk}_i}(\sigma_i, \pi_i) = 1$ and $\sigma_i < c2^\kappa$ **then**
            Add pk_i to $\mathcal{P}_t$
    Compute the Merkle tree root $\mathsf{MRoot}(\mathcal{P}_t)$
    Submit $\mathsf{MRoot}(\mathcal{P}_t)$ to the blockchain
    For each $\mathsf{pk}_i \in \mathcal{P}_t$ **do**
        Send $\mathsf{MProof}(\mathsf{pk}_i \stackrel{?}{\in} \mathcal{P}_t)$ to client i
Step 4 (BH $\ell + \tau$): *Dispute.*
Each qualified client i proceeds as follows
    If i does not receive the proof from the server **then**
        Submit a dispute transaction that consists of $(\sigma_i, \pi_i, \mathsf{pk}_i)$ to blockchain.
Step 5 (BH $\ell + 2\tau$): *Final selection commitment.*
The server proceeds as follows.
    For each dispute transaction $(\sigma_i, \pi_i, \mathsf{pk}_i)$ **do**
        Add pk_i to $\mathcal{P}_f$
    Compute the Merkle tree root $\mathsf{MRoot}(\mathcal{P}_f)$
    Submit $\mathsf{MRoot}(\mathcal{P}_f)$ to the blockchain

We extend the protocol CoSeP to design a protocol CoSeP$_+$ in an active setting. As the active adversary can divert from the protocol, it can instruct the server to exclude some public keys of the qualified clients when computing the Merkle tree

root. Thus, we may not be able to guarantee that all qualified clients are selected. Hence, we add extra steps to the protocol to allow the qualified clients to dispute if the server does not include them in the Merkle tree root.

Further, an active adversary can also try to manipulate the public randomness on the blockchain. Fortunately, by using the VRF, given any randomness, the adversary cannot predict which honest clients are selected. Hence, it cannot affect the selection of honest clients. Plus, based on the security properties of the blockchain, the adversary can only try a bounded number of randomness. Hence, the effect on the selection of colluding clients is bounded. Together, we can guarantee the security of the client selection protocol.

In the registration phase, the server will submit a commitment (i.e., the Merkle tree root) of the client set. As the server may divert from the protocol, the miners will need to verify if the server submits the correct commitment before including the commitment to the blockchain.

As shown in Algorithm 12, protocol CoSeP_+ is extended from protocol CoSeP by adding three new steps: (3) initialize the selection commitment, and (4) dispute and (5) dispute the selection commitment. In step 3, the server composes a list of selected clients and commits the list to the blockchain. In step 4, a qualified client can submit a dispute transaction if s/he is not properly included in the initial list, forcing the server to include her/him in the dispute selection in step 5. By adding those steps, we can guarantee that all honest qualified clients are selected.

Protocol CoSeP_+ is executed in 3τ block heights instead of 2τ block heights. For a training round started at BH ℓ, protocol CoSeP_+ is executed between block heights ℓ and $\ell + 3\tau$. Steps 1 and 2 are the same as in protocol CoSeP.

Step 3: Initial selection commitment. Let $\mathcal{P}_t$ be the set of public keys of qualified clients that are verified by the server. The server submits a selection transaction that

consists of the Merkle tree root $\mathsf{MRoot}(\mathcal{P}_t)$ to the blockchain.

Step 4: Dispute. If a qualified client $i \in \mathcal{U}$ does not receive any Merkle proof from the server, or finds any discrepancy between the Merkle root obtained from the server to the one that the server submitted to the blockchain, it will start a dispute process. The client will submit proof of qualification directly to the blockchain to force its inclusion of itself into the pool. More concretely, at block height $\ell + \tau$, the client can submit a transaction containing the tuple $(\sigma_i, \pi_i, \mathsf{pk}_i)$ to the blockchain. The client i also includes the Merkle proof $\mathsf{MProof}(\mathsf{pk}_i \overset{?}{\in} \mathcal{U})$ to show that its public key is registered.

Step 5: Final selection commitment. At block height $\ell+2\tau$, the server submits a final selection transaction that contains the information of *all dispute transactions*. Let $\mathcal{P}_f$ be the set of the public keys of *dispute clients*, i.e., the clients who submitted dispute transactions. Then, similar to the initial selection, the server constructs a Merkle tree Merkle_f based on $\mathcal{P}_f$. The server submits a final selection transaction that consists of the Merkle tree root $\mathsf{MRoot}(\mathcal{P}_f)$ and sends a Merkle proof $\mathsf{MProof}(\mathsf{pk}_i \overset{?}{\in} \mathcal{P}_f)$ to each client $i \in \mathcal{U}$. Here, before adding the final selection transaction to the blockchain, the miners verify that all public keys of the dispute clients are included in the MRoot_f. The *correctness will be enforced* through smart contracts, executed by all miners in the blockchain.

Proof of membership. We modify the proof of membership in protocol CoSeP to include the selection transaction and dispute selection transaction.

Pool membership verification function. For each client j with the state st_j, the function $\mathsf{PVer}(\mathsf{st}_j, \omega_j^{(i)})$ extracts the blockchain C_j from the local state st_j and then proceeds as follows. The function extract the randomness rnd as in step 1 and verifies $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i, \pi_i) = 1$. It also verifies if the selection transaction and the dispute transaction are included in the header blockchain C_j. If those conditions do not

hold, it returns $\perp$. If all conditions hold, i.e., the proof $\omega_j^{(i)}$ is valid, the function extracts MRoot and MRoot_f from the selection transaction and the dispute transaction, respectively. Then, the function verifies (1) $\sigma_i < c2^\kappa$, and (2) pk_i is included in MRoot_t or in MRoot_f. If those conditions hold, it returns 1, i.e., the client i is selected. Otherwise, the function returns 0, i.e., the client i is not selected.

5.4.2 Security analysis

Pool consistency. In protocol CoSeP, a client is selected if 1) it is qualified, i.e., its VRF output on the randomness is smaller than the given threshold and 2) its public key is either included in the Merkle tree roots of the initial or dispute selection transactions. Similar to the analysis in Subsection 5.3.2, we can show that all honest clients obtain the same value of the randomness. Plus, we can show that the honest clients have the same view on the initial and dispute selection transactions. Indeed, those transaction are included before block height $\ell + 2\tau$. Thus, at the end of the client selection protocol (block height $\ell + 2\tau$), the selection transaction included to the ledgers of the honest clients. Based on persistency property, we can guarantee that all honest clients have the same view on those transactions.

Lemma 93 (Pool consistency)**.** *For any client $i \in \mathcal{U}$, and any honest clients j_1, j_2, we have,*

$$\Pr\left[\exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} \;\middle|\; \begin{array}{l} \mathit{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ \mathit{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array}\right] \leq e^{-\Omega(\kappa)}.$$

Proof. Consider two honest clients j_1, j_2, let $\mathsf{I}(j_1, j_2)$ be the event where the views of

j_1 and j_2 on the set of selected clients are different, i.e.,

$$\Pr[\mathsf{I}(j_1, j_2)] = \Pr\left[\begin{array}{l|l} \exists\ \omega_{j_1}^{(i)}, \omega_{j_2}^{(i)} & \mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1 \wedge \\ & \mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0 \end{array}\right].$$

Assume toward contradiction that there exists a non-negligible number p such that

$$\Pr[\mathsf{I}(j_1, j_2)] > p.$$

Let S be the event where the blockchain protocol is secure, i.e., it achieves chain growth, chain quality, common prefix, persistency, and liveness properties. Similar to proof in Lemma 85, we have,

$$\Pr[\mathsf{I}(j_1, j_2) \mid \mathsf{S}] > p - e^{-\Omega(\kappa)} = p'$$

where p' is a non-negligible number.

Given the event when the blockchain protocol is secure, we consider the case such that there exists $\omega_{j_1}^{(i)}, \omega_{j_2}^{(i)}$ such that $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1$ and $\mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0$.

First, we show that the clients j_1 and j_2 obtain the same selection transaction at the end of the client selection protocol. Indeed, the selection transaction is included at block height $\ell + \tau$. Thus, at the end of the client selection protocol (block height $\ell + 2\tau$), the selection transaction included to the ledgers of j_1 and j_2. Based on the persistence property, the clients j_1 and j_2 have the same view on this transaction. Hence, the selection transaction in $\omega_{j_1}^{(i)}, \omega_{j_2}^{(i)}$ must be the same. Otherwise, either $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)})$ or $\mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)})$ will return $\perp$.

Similar to Lemma 81, given the event S, i.e., the blockchain protocol is secure, we have both j_1 and j_2 obtain the same randomness rnd.

Let MRoot_t be the Merkle tree root in the selection transaction. Let pk_i is the

public key of the client i. Since $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_i) = 1$, pk_i is included in MRoot_t

We obtains (σ_i, π_i) from $\omega_{j_1}^{(i)}$ and (σ_i', π_i') from $\omega_{j_2}^{(i)}$. As $\mathsf{PVer}(\mathsf{st}_{j_1}, \omega_{j_1}^{(i)}) = 1$, we have, $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i, \pi_i) = 1$ and $\sigma_i < c2^\kappa$. As $\mathsf{PVer}(\mathsf{st}_{j_2}, \omega_{j_2}^{(i)}) = 0$, we have, $\mathsf{VRFverify}_{\mathsf{pk}_i}(\mathsf{rnd}, \sigma_i', \pi_i') = 1$ and $\sigma_i' > c2^\kappa$. Thus, we have $\sigma_i \neq \sigma_i'$. This contradict the uniqueness property of the VRF.

□

Pool quality. In contrast with the semi-malicious adversary, the active adversary can try different values of randomness. If a corrupted miner generates the last block of the first τ blocks in a training round, it will compute the randomness value and decide whether or not to publish the block based on the selection of colluding clients. Based on the security properties of the blockchain, we can bound the number of randomness values that the adversary can try. Hence, we can show that the adversary can only slightly increase the chances that colluding clients are qualified. On the other hand, as the adversary cannot predict the VRF outputs of the honest clients, it cannot affect the probability that an honest client is selected.

Since the adversary can also try to exclude the public keys of honest clients, the dispute mechanism in steps 4 and 5 is to ensure that all transactions from honest clients will be included eventually. Thus, our protocol can still achieve the pool quality property in the presence of an active adversary.

Given a randomness rnd, for a client $i \in \mathcal{U}$, let $x_i^{(\mathsf{rnd})}$ be the Bernoulli random variable that represents where or not the client i is qualified, i.e.,

$$x_i^{(\mathsf{rnd})} = \begin{cases} 1 \text{ if } i \text{ is qualified given that } \mathsf{rnd} \text{ is the randomness,} \\ 0 \text{ otherwise.} \end{cases}$$

Follow the same proof as Lemma 86, given a randomness rnd, for any client $i \in \mathcal{U}$,

we have,

$$|\Pr[x_i^{(\mathsf{rnd})} = 1] - c| < e^{\Omega(\kappa)}. \tag{5.3}$$

Plus, the adversary cannot increase or decreases the chance that an honest client get qualified. For any PPT algorithm $\mathcal{A}_R$ that returns a randomness rnd, for any honest client i, we have,

$$|\Pr[\mathsf{rnd} \leftarrow \mathcal{A}_R : x_i^{(\mathsf{rnd})} = 1] - c| < e^{\Omega(\kappa)}.$$

This is given since the VRF outputs of honest clients are pseudorandom and cannot be predicted by the adversary.

We show the pool quality property as follows.

Lemma 94 (Pool quality)**.** *Let $\mathcal{H}$ be the set of honest clients in the set of selected clients $\mathcal{P}$. For $\epsilon > 0$, we have,*

$$\Pr[\frac{\mathcal{H} \cap \mathcal{P}}{\mathcal{P}} \geq \alpha(1-\epsilon)] \geq 1 - e^{-\Omega(nc - \log \kappa)} - e^{-\Omega(\kappa)}.$$

where n is the number of clients, $\alpha = 1 - \beta$ is the fraction of honest clients, and c is the selection probability.

Proof. We prove the pool quality property by bounding the number of honest qualified clients. Then, we will show that all honest qualified clients will be selected. Plus, all selected clients must be qualified. Thus, we can bound the fraction of honest clients.

Let $\mathcal{P}'$ be the set of qualified clients. Let $\mathcal{H}'$ and $\mathcal{M}'$ be the set of honest and colluding clients in $\mathcal{P}'$, respectively.

Consider a randomness rnd', let $\mathcal{M}^{\mathsf{rnd}'}$ be the set of colluding qualified clients with the randomness rnd'. Using the Chernoff bound on the set of $n\beta$ colluding clients $\mathcal{U}'$,

for any $\epsilon'' > 0$, we have,

$$\Pr[|\mathcal{M}^{(\mathsf{rnd}')}| \geq (1+\epsilon'')n\beta c(1+\epsilon')] \leq e^{-\Omega(nc)}.$$

Consider the event where the blockchain protocol is secure. Based on the chain quality property, the adversary can create at most κ blocks among the last blocks used for creating the randomness. Thus, the adversary can try at most κ randomness values. Using the union bound, we have,

$$\begin{aligned}\Pr[|\mathcal{M}'| \geq (1+\epsilon')n\beta c \mid \mathsf{S}] &\leq \kappa e^{-\Omega(nc)} \\ &= e^{-\Omega(nc-\log\kappa)}.\end{aligned}$$

Let rnd be the randomness the honest clients obtained from the step 1 of Algorithm 11. Using the Chernoff bound on the set of $n\alpha$ honest clients $\mathcal{U} \setminus \mathcal{U}'$, for any $\epsilon'' > 0$, we have,

$$\Pr[|\mathcal{H}'| \leq (1-\epsilon')n\alpha c(1-\epsilon')] \leq e^{-\Omega(nc)}.$$

By choosing ϵ such that $\epsilon = 1 - \frac{(1-\epsilon')(1-\epsilon'')}{(1+\epsilon')(1+\epsilon'')}$, we have,

$$\Pr\left[\frac{|\mathcal{H}'|}{|\mathcal{M}'|} \leq \frac{\alpha}{\beta}(1-\epsilon)\right] \leq e^{-\Omega(nc-\log\kappa)}.$$

Consider the event where the blockchain protocol is secure. From the chain quality property, from block height $\ell+\tau$ to block height $\ell+2\tau$, there is at least one honest block that will include all selection transactions. As all honest qualified clients will always submit the selection transactions, all honest qualified clients are selected, i.e., $\mathcal{H} = \mathcal{H}'$

Further, the selected clients must be qualified, i.e., $|\mathcal{P}| \leq |\mathcal{P}'|$. Hence, we have

$\frac{|\mathcal{H}'|}{|\mathcal{P}'|} \geq \frac{|\mathcal{H}|}{|\mathcal{P}|}$. Therefore, we have,

$$\Pr\left[\frac{|\mathcal{H}|}{|\mathcal{P}|} \geq \alpha(1-\epsilon)\right] > 1 - e^{-\Omega(nc\log\kappa)} - e^{-\Omega(\kappa)}.$$

□

Anti-targeting. Since all honest qualified clients are selected, the probability that an honest client is selected is the same as the probability that an honest client is qualified. We can prove the anti-targeting property as follows.

Lemma 95 (Anti-targeting)**.** *Considering an honest client i, for a selection probability c, we have*

$$|\Pr[i \in \mathcal{P}] - c| = e^{-\Omega(\kappa)}.$$

Together, Lemmas 93, 94, and 95 yield the security proof of protocol CoSeP_+ in active settings.

Theorem 96. *The client selection protocol CoSeP_+ achieves pool quality, pool consistency, and anti-targeting properties in the presence of an active adversary.*

5.5 Experiments

We implement prototypes of our proposed protocols, namely SeP, CoSeP, and CoSeP_+, and conduct various experiments to assess their performance in semi-malicious and active adversary settings. The evaluation aims to provide insights into their security properties and performance. From the results, we argue the feasibility and practicality of our solution.

5.5.1 Experimental settings.

The instantiation of the cryptographic blocks is as follows. For the implementation of VRF on clients and server, we use the Libsodium cryptographic library[2]. The VRF verification on the blockchain smart contract is implemented using `vrf-solidity`[3]. As regards the cryptographic hash function, we use the SHA256 compression function which maps inputs of arbitrary length to 256-bit outputs.

Our protocols are evaluated using the following metrics:

- *Error probability*: The upper-bounded error probability of the security guarantees (in Sections 5.3 and 5.4.2).
- *Communication overhead*: The total amount of sent/received data incurred by the protocols.
- *Blockchain cost*: We measure both communication (the total size of the transactions) and computation (the total amount of gas needed to execute the transactions on the blockchain) overhead to maintain the blockchain.

We provide the error probability for each training round and the blockchain cost and the communication overhead for an FL process that consists of 100 training rounds.

All of the experiments in this section are conducted on a single-threaded machine equipped with an Intel(R) Xeon(R) CPU E7-8894 v4 2.40GHz running CentOS. The implementation of the clients and server is in C++ while the blockchain smart contract is in Solidity.

---

[2] `https://github.com/algorand/libsodium/tree/draft-irtf-cfrg-vrf-03`

[3] `https://github.com/witnet/vrf-solidity`

5.5.2 Experiments results

Security properties. Figure 36 shows the impact of the parameter τ on the security error of the three properties: pool consistency, pool quality, and anti-targeting. Later, Figure 37 illustrates the communication overhead of the proposed protocols as a function of τ for both the server and the clients. As can be seen, the parameter τ imposes a trade-off in which higher τ makes the protocol more secure but results in more transmitted data.

Unlike pool consistency and anti-targeting which are invariant to the number of clients, the security of pool quality depends on the number of clients (as analyzed in Sections 5.3 and 5.4.2). Increasing the number of selected clients per round reduces the security error, thereby tightening the guarantee of pool quality.

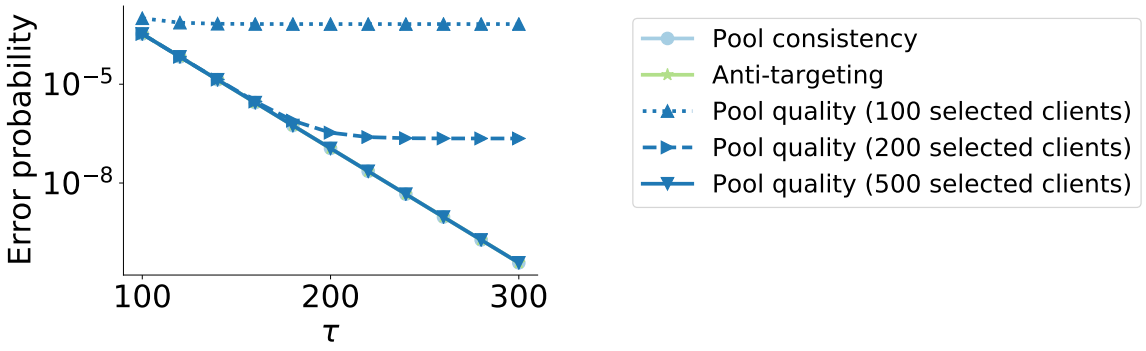
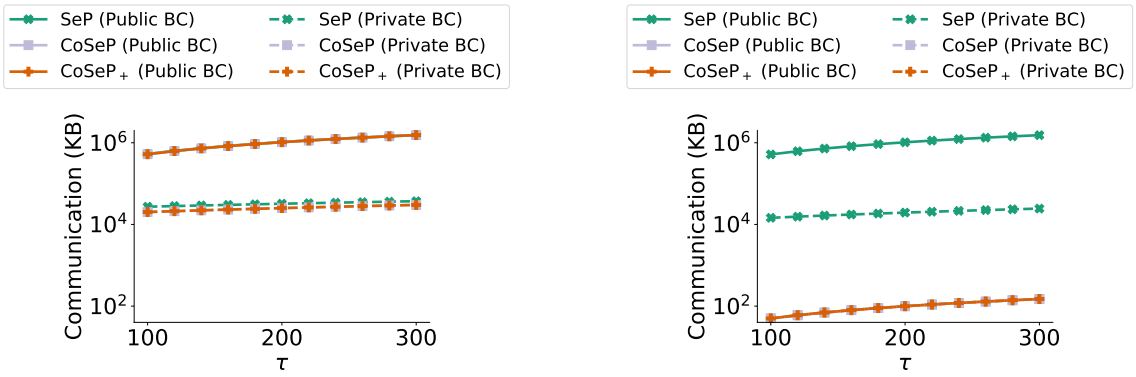


Figure 36: Security error of the protocols on pool consistency, pool quality, and anti-targeting. The result of pool consistency, anti-targeting, and pool quality (with 500 selected client) are identical.

Communication overhead. We measure the communication overhead of the server and each client of the protocols that use *public* or *private* blockchain. A *private blockchain* is dedicated to the FL process, i.e., the blockchain only accepts the transactions from the client selection protocols. On the other hand, the *public blockchain*

can accept other transactions. For the public blockchain, we consider an average block size (excluding the transactions from the client selection protocols) of $52,101$ bytes[4]. We show the communication overhead of the server and each client in Figure 37.



(a) Server. (b) Each client.

Figure 37: Communication overhead of the server and each client. The communication overhead of the server in the protocols that use public blockchains is identical. Regardless of whether public or private blockchain is used, the communication overhead of the client in CoSeP and CoSeP$_+$ are identical.

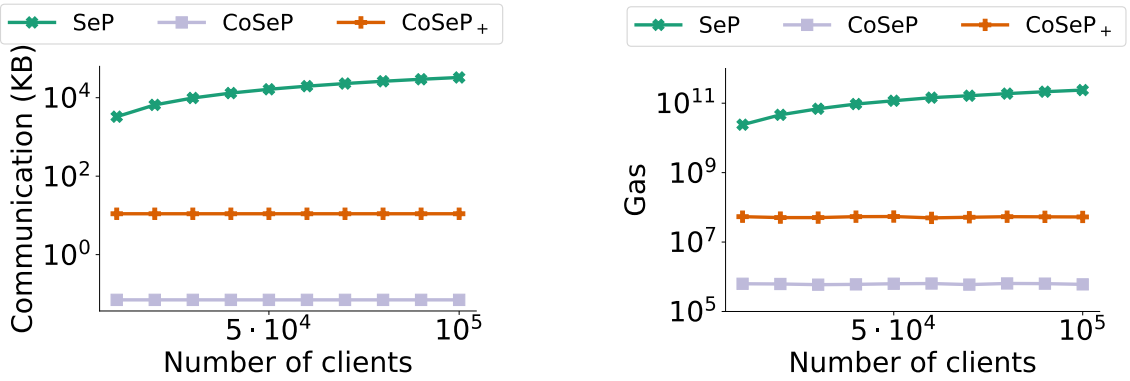
Sever. The communication overhead of the protocols that use the public blockchain are almost identical; since the cost to download the blocks is the dominating factor. For the protocols that use the private blockchain, protocols CoSeP and CoSeP$_+$ have a significantly lower communication overhead, compared to protocol SeP.

Client. The communication overhead of each client in protocol SeP is identical to the server (since the clients also need to maintain for full blockchain). For the protocols CoSeP and CoSeP$_+$, since the clients only need to maintain the lightweight header chain, the communication cost of the client is significantly reduced. Plus, the com-

[4]`https://polygonscan.com/chart/blocksize`. Accessed Nov 15, 2022

munication cost of the client is independent of the size of the blockchain. Specifically, for $\tau = 300$, the communication cost of a client in protocols CoSeP and CoSeP$_+$ is less than 1MB.

Blockchain cost. We compare the size of transactions and gas cost of the blockchain between the protocols SeP, CoSeP, and CoSeP$_+$ in Figure 38. For protocol CoSeP$_+$, we consider the case where there is no dispute client. The protocol CoSeP has the lowest blockchain cost, as it only submits one transaction to the blockchain in the registration phase. Compared with protocol CoSeP, the protocol CoSeP$_+$ has a higher blockchain cost since it has 3 extra steps in each training round to defend against active adversaries. Compared with protocol SeP, the CoSeP$_+$ protocol substantially reduces both the size of transactions and the gas cost, due to its efficiency in communicating with the blockchain. In particular, in CoSeP$_+$ protocol, the total size of transactions for an FL process with 100 training rounds is 14KB.



(a) Blockchain's communication overhead.

(b) Blockchain's computation cost.

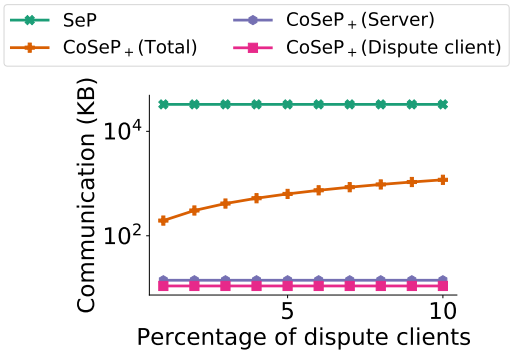
Figure 38: The size of transactions and computation costs on the blockchain for the pool selection.

We also measure the blockchain cost when some clients do the dispute in each training round (see Figure 39). We consider a scenario in which the number of clients

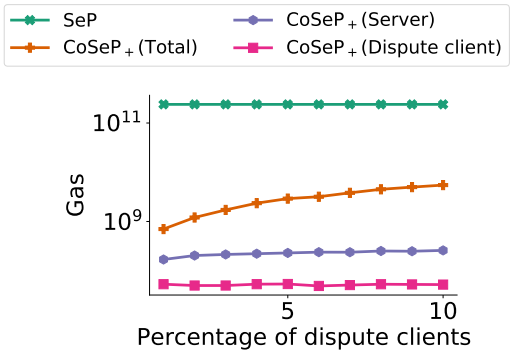
is $100,000$, and we vary the portion of clients submitting disputes from 1% to 10% of the total number of clients. As the portion of clients increase, the blockchain cost for the protocol CoSeP$_+$ also increases. However, comparing with the protocol SeP, the blockchain cost of protocol CoSeP$_+$ is still significantly lower. Specifically, the size of transactions in CoSeP$_+$ protocol 7MB when the dispute clients account for 10% of the total clients. The number in protocol SeP is 31MB.

Blockchain cost in an active setting. In an active setting, the clients in CoSeP$_+$ protocol may not be included in step 3 and need to do the dispute. We show the dispute cost of the CoSeP$_+$ protocol in Figure 39. We consider a scenario in which the number of clients is $100,000$, and we vary the portion of clients submitting disputes from 1% to 10% of the total number of selected clients. As the portion of clients increases, the blockchain cost for the protocol CoSeP$_+$ also increases. However, compared with the protocol SeP, the blockchain cost of protocol CoSeP$_+$ is still significantly lower. Specifically, the size of transactions in CoSeP$_+$ protocol 7MB when the dispute clients account for 10% of the total clients. The number in protocol SeP is 31MB.

Summary. Overall, our proposed CoSeP and CoSeP$_+$ protocols, run fast and generate minimal data (we note that the CoSeP and CoSeP$_+$ protocols are used for semi-malicious and active adversaries, respectively). Specifically, the communication overhead of each client is less than 1MB for an FL process with 100 training rounds. Therefore, it demonstrates that our solution is lightweight for clients and can be used efficiently in practical FL applications. Furthermore, both the blockchain storage and the gas cost are minimal. Thus, our protocols will run efficiently on most blockchain platforms, such as Ethereum and Solana.



(a) Blockchain's communication overhead.



(b) Blockchain's computation cost.

Figure 39: The size of transactions and computation costs on the blockchain for dispute. There is no dispute clients in protocol SeP and CoSeP. We plot their result for comparison.

5.6 Related work

Secure aggregation in FL. With local model updates being the vector for various privacy attacks from the FL server [96, 95], recent research focuses on concealing the local updates from the server. This is achieved via secure aggregation which is a protocol that enables the server to compute the average of the clients' local models without learning anything about each individual local model. Bonawitz et al. [17] is the first work to propose a secure aggregation method from secret sharing and random masking techniques, and then use it to privately aggregate client-provided model updates. Leveraging homomorphic encryption, the authors in [5] and [106] devise a protocol to blindly aggregate the model updates into global models. These secure aggregation protocols can scale up to millions of devices and are robust to clients dropping out. On the other hand, generic SMC protocols that securely compute any function among multiple parties [33, 12, 72] can also be used as secure aggregation in FL. However, they are not scalable enough due to the high complexity of both

computation and communication.

Although these protocols provide strong security guarantees with respect to concealing the local model updates from the server, they assume that the server honestly follows the random client selection, which is not practical. Our paper exposes a new attack vector in the client selection process that can be manipulated by the server to bypass the secure aggregation and learn the local model update of a targeted client. We further devise a verifiable random selection protocol as a countermeasure to prevent the server from conducting such attacks, thereby maintaining the security guarantees of secure aggregation protocols.

Integration of Blockchain and FL. Integrating the immutability and transparency properties of blockchain into FL has been a trending research topic in recent years. Bao et al. [9] propose FLChain which is an auditable and decentralized FL system that can reward honest clients and detect malicious ones. Zhang et al. [108] propose a blockchain-based federated learning approach for IoT device failure detection. Kang et al. [62] develop a reputation management scheme using blockchain to manage and select reliable clients, thereby avoiding unreliable model updates. In [66, 77], the authors utilize blockchain for the exchange and aggregation of local model updates without a central server.

The above-mentioned systems are susceptible to our proposed attack as it does not protect the client selection process from manipulation. Additionally, they are not compatible to be used with a secure aggregation protocol. Our approach to integrating blockchain with FL is different in a way that we use blockchain as a source of randomness for the client selection protocol, such that it enforces the random selection of clients, rendering the client selection attack infeasible.

5.7 Conclusion

In this paper, we have shown that using the secure aggregation protocols alone is not adequate to conceal the local model updates from the server. Through unveiling a new attack vector in the client selection process, we have demonstrated that the server can manipulate the selection to bypass the security guarantees of the secure aggregation protocols and obtain the local updates of clients. We have discussed two strategies to conduct such the proposed client selection attack and analyzed their impact on client privacy. To counter this attack and uphold the security guarantees of secure aggregation protocol, we have proposed a verifiable client selection protocol using blockchain as a source of randomness. As a result, it enforces a random selection of clients in each training round, thereby preventing the server from manipulating the client selection process. We have proven its security against the proposed attack and analyzed its computation cost with Ethereum Solidity to show that it imposes negligible overhead on FL.

Appendix A

ABBREVIATIONS

VCU	Virginia Commonwealth University
PoW	Proof-of-Work
PoS	Proof-of-Stake
FL	Federated learning
SCS	Secure client selection

REFERENCES

[1] https://www.blockchain.com/pools, accessed 11/27/2021.

[2] NXT whitepaper. 2014. https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper_v122_rev4.pdf.

[3] EOS whitepaper. 2018. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on information theory*, 46(4):1204–1216, 2000.

[5] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

[6] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, Oct. 2018.

[7] V. Bagaria, A. Dembo, S. Kannan, S. Oh, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni. Proof-of-stake longest chain protocols: Security vs predictability. *arXiv preprint arXiv:1910.02218*, 2019.

[8] V. K. Bagaria, S. Kannan, D. Tse, G. C. Fanti, and P. Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 585–602. ACM Press, Nov. 2019.

[9] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu. Flchain: A blockchain for auditable federated learning with trust and incentive. In *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 151–159. IEEE, 2019.

[10] S. Basu, I. Eyal, and E. Sirer. Falcon. `https://www.falcon-net.org/`.

[11] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, Aug. 1999.

[12] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351–371. 2019.

[13] I. Bentov, A. Gabizon, and A. Mizrahi. Currencies without proof of work. In *Bitcoin Workshop*, 2016.

[14] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 15–29, 2014.

[15] Bitcointalk. Proof of stake instead of proof of work. July 2011. Online post by QuantumMechanic, available at `https://bitcointalk.org/index.php?topic=27787.0`.

[16] E. Blum, A. Kiayias, C. Moore, S. Quader, and A. Russell. The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In S. Chawla, editor, *31st SODA*, pages 1135–1154. ACM-SIAM, Jan. 2020.

[17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[18] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001.

[19] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[20] V. Buterin. Understanding serenity, part 2: Casper. 2015. `https://blog.ethereum.org/2015/12/28/understanding-serenity-part-2-casper/`.

[21] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, Jan. 2000.

[22] C. L. Canonne. A short note on poisson tail bounds. *Retrieved from the website: http://www. cs. columbia. edu/ ccanonne*, 2017.

[23] J. Chen and S. Micali. Algorand. In *arXiv:1607.01341*, May 2017. `http://arxiv.org/abs/1607.01341`.

[24] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou. Utility maximization in peer-to-peer systems. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):169–180, 2008.

[25] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al. Op-

portunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[26] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan. Can network coding help in p2p networks? In *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–5. IEEE, 2006.

[27] K.-M. Chung, H. Lam, Z. Liu, and M. Mitzenmacher. Chernoff-hoeffding bounds for markov chains: Generalized and simplified. *arXiv preprint arXiv:1201.0559*, 2012.

[28] M. Corallo. Bitcoin relay network.

[29] M. Corallo. Compact block relay. `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`.

[30] S. Coretti, A. Kiayias, C. Moore, and A. Russell. The generals scuttlebutt: Byzantine-resilient gossip protocols. *CCS*, 2022:541, 2022.

[31] R. Dahlberg, T. Pulls, and R. Peeters. Efficient sparse merkle trees. In *Nordic Conference on Secure IT Systems*, pages 199–215. Springer, 2016.

[32] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In I. Goldberg and T. Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 23–41. Springer, Heidelberg, Feb. 2019.

[33] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[34] B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, Apr. / May 2018.

[35] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[36] S. Deb, S. Kannan, and D. Tse. Posat: Proof-of-work availability and unpredictability, without the work. *FC 2021,*, 2020.

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

[38] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni. Everything is a race and nakamoto always wins. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 20*, pages 859–878. ACM Press, Nov. 2020.

[39] J. Dillon. Bitcoin-development mailinglist: Protecting bitcoin against network-wide dos attack, 2018.

[40] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, Jan. 2005.

[41] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs

and keys. In *International Workshop on Public Key Cryptography*, pages 416–431. Springer, 2005.

[42] D. Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.

[43] R. Dorfman. A formula for the gini coefficient. *The review of economics and statistics*, pages 146–149, 1979.

[44] L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2021.

[45] A. Frieze and M. Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

[46] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer, 2015.

[47] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, Apr. 2015.

[48] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[49] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer. Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998*, 2018.

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

[51] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017. `https://eprint.iacr.org/2017/454`.

[52] M. Goemans. Chernoff bounds, and some applications. *`https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf`*, 2015.

[53] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[54] S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, Aug. 1993.

[55] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

[56] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In J. Jung and T. Holz, editors, *USENIX Security 2015*, pages 129–144. USENIX Association, Aug. 2015.

[57] C. C. Heyde. On a property of the lognormal distribution. *Journal of the Royal Statistical Society: Series B (Methodological)*, 25(2):392–393, 1963.

[58] F. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson. Blockchain-based e-voting system. In *2018 IEEE 11th international conference on cloud computing (CLOUD)*, pages 983–986. IEEE, 2018.

[59] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[60] J. Huang, D. He, M. S. Obaidat, P. Vijayakumar, M. Luo, and K.-K. R. Choo. The application of the blockchain technology in voting systems: A review. *ACM Computing Surveys (CSUR)*, 54(3):1–28, 2021.

[61] J. R. Jensen, V. von Wachter, and O. Ross. An introduction to decentralized finance (defi). *Complex Systems Informatics and Modeling Quarterly*, (26):46–54, 2021.

[62] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[63] A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. `https://eprint.iacr.org/2015/1019`.

[64] A. Kiayias, S. Quader, and A. Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. *arXiv preprint arXiv:2001.06403*, 2020.

[65] L. Kiffer, R. Rajaraman, and a. shelat. A better method to analyze blockchain consistency. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 729–744. ACM Press, Oct. 2018.

[66] H. Kim, J. Park, M. Bennis, and S.-L. Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019.

[67] V. King and J. Saia. From almost everywhere to everywhere: Byzantine agreement with õ(n3/2) bits. In *International Symposium on Distributed Computing*, pages 464–478. Springer, 2009.

[68] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for p2p streaming systems. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 919–927. IEEE, 2007.

[69] R. Kumar and K. W. Ross. Peer-assisted file distribution: The minimum distribution time. In *Hot Topics in Web Systems and Technologies, 2006. HOTWEB'06. 1st IEEE Workshop on*, pages 1–11. IEEE, 2006.

[70] J. Kwon. Tendermint: Consensus without mining. 2014. `https://tendermint.com/static/docs/tendermint.pdf`.

[71] J. Li, P. A. Chou, and C. Zhang. Mutualcast: An efficient mechanism for one-to-many content distribution. In *ACM Sigcomm Asia Workshop*. Citeseer, 2005.

[72] Y. Lindell, B. Pinkas, N. P. Smart, and A. Yanai. Efficient constant round multi-party computation combining bmr and spdz. In *Annual Cryptology Conference*, pages 319–338. Springer, 2015.

[73] C.-D. Liu-Zhang, C. Matt, U. Maurer, G. Rito, and S. E. Thomsen. Practical provably secure flooding for blockchains. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9,*

2022, Proceedings, Part I, pages 774–805. Springer, 2023.

[74] C.-D. Liu-Zhang, C. Matt, and S. E. Thomsen. Asymptotically optimal message dissemination with applications to blockchains. *Cryptology ePrint Archive*, 2022.

[75] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Transactions on information theory*, 52(6):2608–2623, 2006.

[76] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.

[77] C. Ma, J. Li, M. Ding, L. Shi, T. Wang, Z. Han, and H. V. Poor. When federated learning meets blockchain: A new distributed learning paradigm. *arXiv preprint arXiv:2009.09338*, 2020.

[78] Y. Marcus, E. Heilman, and S. Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. *IACR Cryptol. ePrint Arch.*, 2018:236, 2018.

[79] C. Matt, J. B. Nielsen, and S. E. Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, pages 400–430. Springer, 2022.

[80] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[81] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.

[82] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. `https://bitcoin.org/bitcoin.pdf`.

[83] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[84] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh. Erlay: Efficient transaction relay for bitcoin. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 817–831. ACM Press, Nov. 2019.

[85] G. Naumenko, G. Maxwell, P. Wuille, S. Fedorova, and I. Beschastnikh. Bandwidth-efficient transaction relay for bitcoin. *arXiv preprint arXiv:1905.10518*, 2019.

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

[87] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[88] R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, Apr. / May 2017.

[89] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[90] R. Pass and E. Shi. The sleepy model of consensus. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Heidelberg, Dec. 2017.

[91] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[92] P. R. Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.

[93] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen. Revisiting nakamoto consensus in asynchronous networks: A comprehensive analysis of bitcoin safety and chainquality. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 988–1005, 2021.

[94] M. Saad and D. Mohaisen. Three birds with one stone: Efficient partitioning attacks on interdependent cryptocurrency networks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1404–1418. IEEE Computer Society, 2022.

[95] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[96] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security*

and Privacy (SP), pages 3–18. IEEE, 2017.

[97] R. Sweha. Angels: In-network support for minimum distribution time in p2p overlays. Technical report, Boston University Computer Science Department, 2009.

[98] M. Takemiya and B. Vanieiev. Sora identity: Secure, digital identity on the blockchain. In *2018 ieee 42nd annual computer software and applications conference (compsac)*, volume 2, pages 582–587. IEEE, 2018.

[99] P. Vasin. Blackcoin's proof-of-stake protocol v2. 2014. `http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf`.

[100] Y. Wang and R. Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 178–188, 2020.

[101] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*, 2021.

[102] G. Wolfond. A blockchain ecosystem for digital identity: improving service delivery in canada's public and private sectors. *Technology Innovation Management Review*, 7(10), 2017.

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

[104] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham. HotStuff: BFT consensus with linearity and responsiveness. In P. Robinson and F. Ellen, editors, *38th ACM PODC*, pages 347–356. ACM, July / Aug. 2019.

[105] D. A. Zetzsche, D. W. Arner, and R. P. Buckley. Decentralized finance. *Journal of Financial Regulation*, 6(2):172–203, 2020.

[106] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 493–506, 2020.

[107] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr. Federated learning for internet of things. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 413–419, 2021.

[108] W. Zhang, Q. Lu, Q. Yu, Z. Li, Y. Liu, S. K. Lo, S. Chen, X. Xu, and L. Zhu. Blockchain-based federated learning for device failure detection in industrial iot. *IEEE Internet of Things Journal*, 8(7):5926–5937, 2020.

[109] H. Zhong, Y. Sang, Y. Zhang, and Z. Xi. Secure multi-party computation on blockchain: An overview. In *Parallel Architectures, Algorithms and Programming: 10th International Symposium, PAAP 2019, Guangzhou, China, December 12–14, 2019, Revised Selected Papers 10*, pages 452–460. Springer, 2020.

VITA

Phuc Thai received his BS in Computer Science from Vietnam National University in 2016. He joined the Doctor of Philosophy program at Virginia Commonwealth University, Richmond, Virginia in 2018. He is currently working as a research assistant under the supervision of Dr. Thang Dinh and Dr. Hong-Sheng Zhou. His research interests are in developing secure blockchain protocols and its applications.