Blockchain: Ecosystem, Security and Performance

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Shoeb Siddiqui
2018701017
shoeb.siddiqui@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2021

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Blockchain: Ecosystem, Security and Performance" by Shoeb Siddiqui, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. Sujit Gujar

To Humanity's Knowledge,
Understanding and Progress

Acknowledgments

I thank

- Prof Sujit Gujar, for his guidance and support
- Ganesh Vanahalli, for his part in my work
- My lab-mates, colleagues, and teachers, for all their help and assistance

Abstract

Blockchain technology has been one of the most disruptive innovations in the last decade. Blockchain is a decentralized, append-only trustworthy distributed database. It was invented to build Bitcoin – to democratize financial systems, in particular, currency. However, soon the researchers learned the power of blockchain to build smart contracts, programs that can execute and validate contracts, which otherwise were traditionally done through legal contracts. Since then, there are a plethora of applications has been envisioned using blockchain.

At the backend, miners are the ones who maintain the blockchain in anticipation of fair rewards. Bitcoin offers two types of rewards for the miners: (i) block rewards that halve every four years and (ii) transaction fees. Halving block rewards is done by making currency inflation-free. Transaction fees are optional. This can lead to stagnation of transactions and unprofitability for miners. Another challenge with Bitcoin or many other blockchains, to build real-world applications, is performance. Many times, improving performance in blockchain leads to different security issues.

In this thesis, we address both research challenges: (C1), how to configure the Bitcoin reward structure to maintain its ecosystem healthily as well as inflation-free. (C2), how to design a blockchain that is performant as well as secure. Towards C1, we propose BitcoinF, a simple modification to Bitcoin rewards that ensures fairness to the miners. We analyze it game theoretically. Towards C2, we propose QuickSync, a novel blockchain protocol that is faster than most of the secure blockchain protocols such as Bitcoin and Ourborous and retains all the security properties. We do security analysis for different attacks such as Sybil attacks and prove the robustness of QuickSync towards the attacks we studied.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Transactions and their records form the core of economy. Both currency and non-fungible assets require to be handled by transactions, and are addressed by payment/banking mechanisms and contracts respectively. The record of these transactions have a defining say in validity, quality and quantity. It is due to this, that the one who proposes modifications and maintains the record, has control over the *economic ecosystem*.

Traditional economic and banking systems entrust the maintenance of these transaction records to a centralized authority. Due to the high stakes involved these centralized authorities have an incentive to act selfishly without regard for its users. The space of such behaviours in traditional systems are fairly restricted by the use of *cryptography*, i.e, a digitally signed transaction itself cannot be mutated or denied by the signer. However, unjust *denial of service*, unjust confiscation or modification of assets, and manipulating the timeline/sequence of events, are some of the things that malicious central authorities can still do.

To address these issues of trust we can leverage *blockchains*. Blockchain technology was introduced in *Bitcoin* [55], it is one of the most significant technological innovations of this century. A blockchain is an append-only, secure, transparent, distributed ledger. It stores the data in *blocks* connected through immutable cryptographic links, with each block extending exactly one previous block. Blockchains inherently provide an immutable sequence of transactions. Further, blockchains enable *decentralization*, thus disabling unilateral control preventing prevent unjust actions. The underlying technical problem that blockchain solves is *byzantine fault tolerant distributed consensus* on a database in a decentralized system. Byzantine fault tolerant distributed consensus, means to achieve agreement amongst the distributed instances of execution while resisting the efforts of adversarial entities.

A traditional centralized system, is typically built as a *client-server model*. The client, controlled by the user, sends request to the server, controlled by the central authorities, which then processes the request as it pleases. The *ecosystem* and the *incentive mechanism* of such a system is rather simple as the centralized authority acts a universal fulcrum on which every process can wholly rely on. While

such systems are performant and efficient, they require the central authorities to be trusted implicitly. A decentralized blockchain, has a *protocol* for all nodes to follow. Most nodes are expected to follow this protocol while some nodes are expected to deviate *rationally* or *maliciously*. The ecosystems of such systems are very complex and must be configured correctly to ensure that the blockchain functions as intended. In addition to the complexities of the ecosystem, such systems typically face performance, security and efficiency challenges. In spite of these challenges blockchains have been rigorously studied and applied to a myriad of use cases that involve establishing *distributed trust*.

A key driving factor in establishing blockchains as mainstream technology rather than a passing fad has been the research and study that scrutinize and improve blockchain technology. One such domain is *game theory*. It provides the necessary framework to model and analyze decentralized ecosystems and their incentive mechanism with rational participants. It very useful in studying the vulnerabilities of a blockchain in various scenarios, and helps translate that understanding into actionable information be it setting the boundaries of operation or extending them. To comprehensively represent these ideas of favourable conditions that lead to the intended function, we use the concept of the *health of a blockchain*. It is a composition of a set of quantitative metrics. The health of a blockchain and the *fairness* experienced by the participating entities, miners and users, are not only co-related but also casually linked. This is primarily due the role that incentive mechanisms play in the blockchain ecosystem. Since *altruism* is unrealistic, rational participants require monetary incentives to motivate them to act appropriately. These incentives come in various shapes and forms.

Another crucial domain of research is *chain construction*. Since the participants of a blockchain must coordinate their efforts, even the otherwise simple task of constructing the chain by extending it step-by-step becomes a challenging one. The approach a blockchain protocol takes to construct its chain is what determines the security and performance of the blockchain. As more and more blockchain protocols are developed and marketed, differentiating factors between them become the key to their success. Performance is one such key differentiator; every blockchain protocol would ideally like to have as high performance as possible, but of course this has limits. The performance of a blockchain is fundamentally limited by its security characteristics. One can not increase the one of them without sacrificing on the other. However, new blockchain protocols, might use drastically different techniques for chain construction and hence may have completely different security-performance trade offs. Thus, the two domains of research, game theory and chain construction, address two major challenges of blockchain technology:

C1 Configuring the blockchain ecosystem

C2 Improving chain construction mechanics

C1: The blockchain ecosystem typically have one major concern: *inflation*. To control inflation, Bitcoin halves the *block reward* every four years, so that the total amount of Bitcoin ever mined does not exceed a certain value. Thus, in Bitcoin, there would come a scenario when the block rewards would

be insignificant. In such a scenario the miners would then have to rely on the *transaction fees*. Since this is an inevitable scenario, its implications must be studied and any adverse effects must be resolved.

C2: The first blockchain Bitcoin, used a *Proof-of-Work* (PoW) mechanism (with a longest chain rule) for chain construction. This PoW mechanism is highly in-efficient in terms of power consumption, and is severely limited in terms of performance. To be able to use blockchains practically on a large scale, the blockchain's performance must be improved while ensuring the security of the blockchain.

A blockchain must have a well conditioned ecosystem, be secure against byzantine adversaries and competitively performant, for it to be a viable endeavour for people to participate and invest in.

1.2 Contributions

Our contributions are two-fold. To address C1, we propose *BitcoinF* and for C2, we propose *QuickSync*. Quick summary of these contributions is presented below.

1.2.1 *BitcoinF*: Achieving Fairness for Bitcoin in Transaction-Fee-Only Model

We consider *Bitcoin* under what we term as *mature operating conditions* (MOC), i.e., when block rewards are insignificant, *transaction-fee-only model* (TFOM), and the influx of transactions is equal to the processing capacity, *standard influx*. We show that in *Bitcoin* under MOC, *miners* following *first-in-first-out* (FIFO) processing take heavy losses as compared to the ones mining *greedily*. This is an issue, as miners depend entirely on transaction fees to sustain mining and cannot take considerable losses in order to follow FIFO processing. This results in all miners processing transactions greedily. As we show in our analysis, this causes transactions to experience unreasonably high processing latency; such transactions are referred to as *stranded transactions*. This leaves the *users* with uncertainty about whether or not their transactions will be processed. A compounding effect of stranded transactions is the rising *price of consumption*. These issues culminate in unfairness for both the miners and the users. Thus, we say that Bitcoin, in its current form, is unfair under MOC (Proposition 1).

We solve these issues of unfairness by proposing a novel protocol, *BitcoinF*, for processing transactions. *BitcoinF* enforces a minimum transaction fee and uses two queues, instead of one to process transactions. *BitcoinF* allows the users to express urgency and have their transactions prioritized, just as they can do in Bitcoin. Our game-theoretic analysis proves that the proposed modification to Bitcoin, *BitcoinF*, ensures good health of the blockchain. Thus, we believe *BitcoinF* will lead to a stable ecosystem and hence will be fair to the miners and the users (Proposition 2). While there have been many published works analyzing TFOM, using collected data or using game-theoretic models, to the best of our knowledge, this is the first formal attempt at solving the pressing issues that are bound to arise in TFOM.

1.2.2 *QuickSync*: A Quickly Synchronizing PoS-Based Blockchain Protocol

We propose a novel *Proof-of-Stake* (PoS) blockchain protocol, *QuickSync* that is secure against *fully adaptive corruptions* (FACs), and achieves slightly better tps, and improves on the t_f by a factor of 3, as compared to *Ouroboros v1*. Essentially, it quickly synchronizes (resolves) the *forks* that arise. To build *QuickSync*, we employ the framework of the Ouroboros protocol. *QuickSync* differs from Ouroboros v1 and Ouroboros Praos, in both aspects of chain construction, the *block publisher selection mechanism* (BPSM) and the *chain selection rule* (CSR). The key idea is, we propose a metric called *block power*, assigned to each block. With this, we compute *chain power* of an every competing chain. The chain power of a chain is the sum of block powers of all the blocks of the chain. Using chain powers, we establish the *best chain*, the one with the highest value for this metric, that the node must build on from a given set of chains. It results in ensuring that all forks are trivially resolved, except for the ones generated by the adversary. Block power is a function of the output of the node's privately computed *Verifiable Random Function* (VRF) output based on the *digital signature* of the node, *seed randomness*, and the *slot counter*. The VRF output is not revealed until the block is published, thus enabling immunity against FACs. Block power is also a function of the *relative stake* that the node holds, to enable the PoS aspect of the mechanism. A naive implementation of such a concept is vulnerable to *Sybil attacks*. To solve this, we present a *Sybil attack resistant function*, which we utilize for the block power metric, with the help of a technique called *histogram matching*. As multiple nodes publish blocks at the same time, it may seem that there will be several forks in *QuickSync*, as is the case in the other PoS protocols such as Praos. The key novelty here is that we resolve these forks immediately using block power rather than relying on the network to eventually resolve them using the longest chain CSR. Thus, *QuickSync* is in sharp contrast to other PoS-based mechanisms that *do not differentiate* between the published blocks.

1.3 Overview

The rest of the thesis is structured as follows. In Chapter 2, we discuss the preliminary concepts required to appreciate blockchain technology. In Chapter 3, we present our work, BitcoinF, on the Bitcoin ecosystem. In the next chapter, we present our work, QuickSync, on the security and performance of Proof-of-Stake protocols. We conclude the thesis in Chapter 5.

Chapter 2

Preliminary Concepts

2.1 Blockchain

As mentioned in the introduction a blockchain is an append-only, secure, transparent, distributed ledger. In this section we will explain how blockchain achieves these properties and how they are key in the blockchains usefulness.

At its most basic a blockchain is just a series of blocks of data, just like any other sequenced batched data, except for one difference, *hash links*. Consider a traditional database which keeps sequenced batches of data as record. In such a traditional database, the data stored can be modified by the authority managing them without consequence. This modification can be a change in sequence or a removal or addition, and are possible even with the use of cryptographic techniques such as *digital signatures*. The authority has this ability as each piece of data is an isolated entity, independent of another, in the database. This causes any undisclosed modification to go unnoticed unless that specific piece of data is somehow compared with a previous record, which is impractical given that traditional databases are controlled in totality by single entity that can choose to not share the records due to claimed privacy or security concerns.

A blockchain gets rid of these issues with hash links. Hashes are succinct representations of data as a many-to-one mapping on a reduced (yet still astronomically large) output space. These hashes are deterministic and hence can be verified against the data but themselves do not reveal any information about the data itself. Changing any part of the data in any way would result in an unpredictable change in hash. While there may be collisions, i.e., two data sharing the same hash, is incredibly rare. In a blockchain, these hashes are used to represent blocks of data. To create links between blocks the hash (succinct representation) of a block is added to the next block. This hash of the block that is included in the next block is used in computing the hash of the next block (and included in the one after), thus creating a chain of hashes. Since hashes register any change in data, if the data in a block is changed in any way, its hash will change and hence the hash of the next block will change. So any change in data of any block will create a rippling change in the hash of every subsequent block. Thus, to monitor the integrity of the entire database only the hash of the last block is required, which reveals nothing

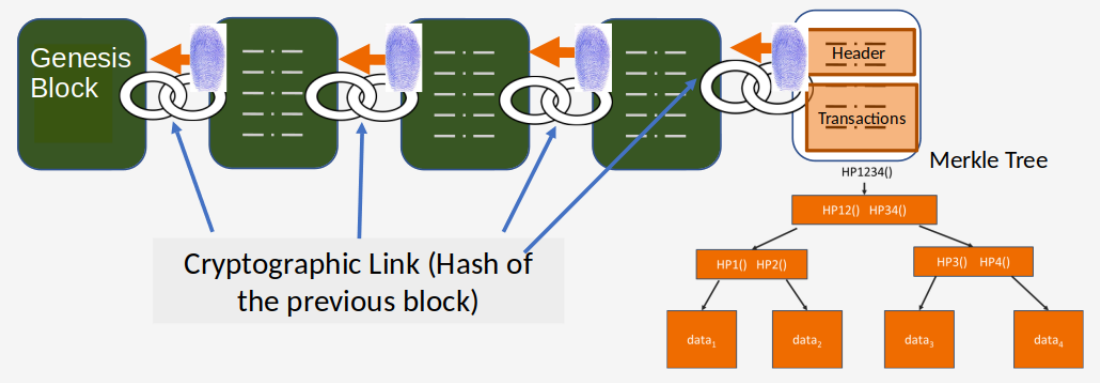


Figure 2.1: Structure of a blockchain

about the data. Thus, given the hash of the last block to verify the database against, we can say that the data record in the database is immutable until the last block. The only modification that be made to the blockchain once the hash of the currently last block is given, is to add blocks to the end of the chain extending it. This establishes the secure and append-only nature of the blockchain.

Another issue with traditional systems is that even if such modifications get noticed by the public, there is nothing that can really be done about them without extensive legalities, that is if the legal framework for it even exists. Besides the ability to tamper with the data maliciously this ability of taking unilateral decisions extends to governance as well. In other words, the authorities in charge of the traditional system can more or less act as they like with impunity. They do not share the decision making power with anyone else as they control the data entirely. To invest in such a system in any way requires trusting the authority in control. Given the large stakes involved the authority always has a motive to act maliciously, this not only causes mistrust, but also has resulted in countless unfortunate incidents in the history of centralized control.

Blockchain solves this by enabling *decentralization*. Even before blockchains there existed algorithms to achieve *byzantine fault tolerant consensus*, but these could never be applied to practical decentralized scenarios as it is computationally and communicationally infeasible to apply such algorithms to entire databases; which is what would be required, as consensus on batches of data containing incremental changes would be insufficient without the immutable links between batches of data. This is precisely what blockchains solve about decentralization, instead of needing to achieve consensus on the entire database, the hash links between blocks only require that each incrementally added block be agreed upon. The culmination of the ideas of byzantine fault tolerance and blockchain resulted in *Bitcoin*, the first practical use of what we now know as blockchain technology. Since in a decentralized system, the execution is distributed amongst all the participants and hence so is the data, this makes blockchain a *transparent, trust-worthy, append-only distributed ledger*.

Due to such interesting properties, blockchains have found widespread applications not only in financial domains, but also in many industry, enterprise, e-governance applications to name a few, e.g. [15, 18, 19, 54] and the references cited therein.

2.2 Structure of a Blockchain

A blockchain is just a series of blocks of data that reference each other. However, to use a blockchain in a decentralized system, it must be self-defining, as its interpretation cannot be offloaded to an externality. To this end, blockchains have structure, and quite similar ones too. Most all blockchains have four defining structural characteristics: 1) Genesis block, 2) Block header, 3) Block data, 4) Hash Links.

Genesis Block The Genesis Block is the first block in the blockchain. It defines the blockchain itself, by defining the its various aspects either implicitly or explicitly. It determines the type of blockchain, its identification, and its parameters. This is the block that differentiates between blockchains. The exact structure and definition of a genesis block varies from blockchain to blockchain, but its purpose is to provide a unique starting point for the blockchain. The genesis block forms the origin (root) of the chain (tree) of blocks. All blocks that are a part of a particular blockchain extend the chain originating from the corresponding genesis block.

Block Header The Block Header is the part of the block that contains the metadata pertaining to the block. The block header of a block is what defines the block. It contains information that, identifies it as a part of a certain blockchain, determines it place in the blockchain, validates it as a valid block, and may contain auxiliary information such as timestamps. It contains the hash of the previous block, this is what links the block header and hence the block to its predecessor. It contains the *merkle root*, which is a succinct representation much like a hash, of the block data, this is what links a block header to its block data.

Block Data The Block Data is what the blockchain is built around. The Block Data is simply a sequence records, commonly called *transactions*. Structurally this is the simplest part of the blockchain. Its integrity is validated by its merkle root contained in the block header. The block data is typically limited in size to account for communication and network limitations.

Hash Links Hash Links are what connects blocks to their predecessors. They are used to validate the block's placement in the blockchain, using its predecessor. They are what makes a blockchain an immutable chain rather than an unordered set of blocks.

2.3 Blockchain Network

A blockchain network is typically composed of two kinds of participants: *miners* and *users*. Miners also called *nodes*, are those who run and maintain the blockchain. Users are those who utilize the blockchain and its services. The miners communicate with each other to execute the blockchain. Since it is infeasible for a miner to connect with every other miner in the network, miners connect to a subset

of miners. These connections between miners form a connected graph. The miners share relevant information with the nodes they are connected to, this information is further propagated throughout the network, this process is typically referred to as *gossiping*.

Miners form the basis of the network necessary for the blockchain to function. Users connect to miners to interact with the network and hence the blockchain. Miners may come in shapes and sizes vis-a-vis their responsibilities (at a given point in time), their commitment and investment level, and their capacities.

An important distinction between blockchain networks is whether they or *permissioned* or *permissionless*. In permissioned blockchains, there are roles assigned to participants and no one is allowed to act beyond the scope of their role, moreover external entities can not participate in the network without prior approval. In permissionless blockchains, any one can become a miner or a user, external entities aren't restricted to begin participating in the network.

2.4 Cryptography in Blockchains

Cryptography is fundamental in making blockchains possible, is it ubiquitous in the blockchain space. Blockchains rely on cryptography for integrity, authenticity, obscurity and unpredictability. Blockchain leverages cryptographic key pairs and digital signatures for *pseudonymous* authentication. Integrity and obscurity are achieved by hashes. Unpredictability is a key requirement in a decentralized system with opportunistic entities that would leverage the knowledge of outcomes for an unfair advantage. To achieve unpredictability, blockchains use randomness as inputs to deterministic functions. Multiparty cryptograhpic computations can be used to generate unbaised unpredictable randomness using a *random seed*, a process that can be chained.

2.5 Blockchain Database

The Blockchain Database comprises of the *transaction log* and the *world state*. The transaction log is the entire history of the blockchain, whereas, the world state is the currently relevant snapshot, of the blockchain useful for efficient processing of transactions. The transactions in the block data, in sequence, form the transaction log. The world state is a collection of storage items in their current state. When a block is added to the chain, the transactions in the block data are sequentially processed to avoid unpredictable non-deterministic race conditions. The transactions may refer to and modify the world state storage objects, causing *state transitions*.

Typically the transaction log doesn't serve much purpose and has a very high storage footprint. This can be addressed by block pruning. Once a certain block has been established with extremely high confidence by the entire network, data before it can be discarded. However, this has to be done intelligently to avoid discarding data that is still relevant to the current state of that particular blockchain.

2.6 Blockchain Runtime

Most modern blockchains not only support storing and modifying storage objects, but also allow program code to be uploaded as a transaction. This program code is commonly known as *smart contract*. These smart contracts expose functions, which allow for complex inputs, processing and output. The sheer complexity of executing smart contracts feasibly require a high level abstraction called *virtual machines* or *runtimes*. These runtimes emulate a full fledged *Turing complete* machine. However, Turing complete machines are subject to the *Halting problem*, wherein it is not possible to know if and when the execution of a certain code will terminate. To address this problem, most blockchains use some form of computation metering to prevent the the code from running for too long, to determine the computation capacity that needs to be set aside for it and the cost that would be need to be paid for the execution. Another solution is to simply use a *non-Turing complete* runtime, but this greatly limits the capability of the runtime.

2.7 Blockchain Consensus

To achieve decentralization, a single entity must not be allowed to control execution. This is truly feasible only when participants use their own resources *independently*. This applies to blockchains as well. To leverage blockchain for decentralization, the execution of the blockchain must be distributed. This means that each miner must use their own hardware and execute the blockchain by itself given *inputs* for the execution to generate blocks to be added to the blockchain as *outputs*. Inputs for execution for a blockchain miner is primarily the information communicated to him by mining peers regarding transactions received to be included and the chain to extend. Since there is no external authority that determines these execution inputs, the miners may receive, via gossiping, differing information based on what miners that he is connected to have heard and what they share. Due to this differing input information across miners, miners may have a different output of execution, even though the execution itself is deterministic. In addition to this, personal decisions on how to use the input information, such as which transactions to include or which chain to extend, further add to the difference in the resulting output of the individual execution. This is essentially, the problem of different miners proposing different blocks to be added to the blockchain. If miners do not agree on which block is to be added then the record will not remain consistent across miners, and they will disagree on the state of the storage objects. Such disagreement, is, of course, is a major issue when dealing with databases and records of any kind. To address this issue the miners must come to an agreement on which block to add to the blockchain, this is what it means to have consensus. Consensus of miners on block addition is what keeps the blockchain consistent. Consensus on blocks is a major aspect of a blockchain protocol.

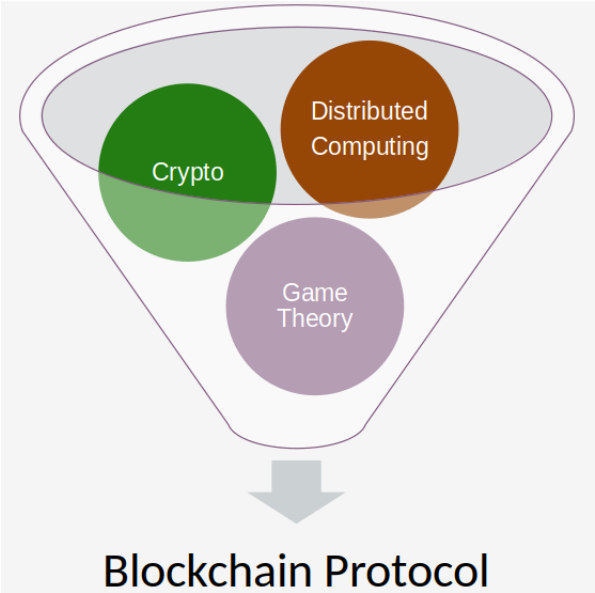


Figure 2.2: Key Elements of a Blockchain Protocol

2.8 Blockchain Protocol

A *protocol* is a set of steps that describe an execution. To execute any process the executor must follow a protocol. A distributed process requires a common protocol for all executors to follow, as it ensures compatibility. Compatibility is essential for processes in which the outputs of the execution is further used as its inputs, such as in a blockchain; eg: In a blockchain one of the outputs of execution are the blocks that miners produce which are then communicated to the rest of the network and are used by the other miners to extend the chain upon. In blockchains incompatibility leads to network fragmentation and thus potentially disagreement with the data record. A blockchain protocol determines and ties together the all the key aspects of the blockchains and its execution. It is what determines the underlying technologies and their specific implementation. A blockchain is effectively an instance of a blockchain protocol, where the genesis block of the blockchain differentiates between instances. A blockchain protocol induces the blockchain's ecosystem and determines its security and performance characteristics.

2.9 Blockchain Ecosystem

The blockchain ecosystem refers to interactions between the participants of the blockchain in the context of the blockchain. The blockchain ecosystem depends on the behavioral inclination of the participants, what they are looking to gain from their efforts and investment, what are the triggers they respond to. Typically, money is the primary consideration of the participants and is hence one of the most important aspects of the blockchain ecosystem. The monetary aspects of a blockchain involve,

the *incentive mechanism*, *penalties* and *cost structure* for miners, and the *fee structure* for users. This is considered by the miners to determine whether or not they should mine, how much resources to invest in mining and whether they should *deviate* from the protocol; and by the users to determine if they should transact on the blockchain and if so, how much should they pay for the service. The *fairness* experienced by the participants determines the whether or not they will actively interact with the blockchain, while the health of the blockchain determines whether or not they should keep their investment or forego it. Thus, different types of miners and users compose ecosystem, with different approaches to the duties they carry out and with different levels of monetary and resource commitments. So why is it important to be concerned about the blockchain ecosystem? The reason is ultimately that we require the miners and users to continue their, participation in and interaction with, the blockchain. This is because miners help run an maintain the blockchain, while users add value to it. The miners and users will stop participating if they do not experience fairness. Further, they will withdraw their investment if they see the health of the blockchain drop, because deterioration of the blockchain's health signals a drop in value of the blockchain. One way that the value of the blockchain drops is if the participants do not experience fairness, and hence the metrics that quantify root cause of fairness form a part of the health of the blockchain. One highly significant way in which the health of the blockchain is adversely affected is if the participants deviate from the prescribed protocol. They might do this for a monetary gain. Due to this reason it is important for the protocol to ensure that the desired behavior is sustainable with rational participants, in other words, that it is *strategic equilibrium*.

2.10 Blockchain Security and Performance

A blockchain protocol must protect against malicious entities trying to subvert it for their own rational or irrational reasons, while being performant enough to be used in practical real world applications. The security and performance of blockchains must be discussed together as there are trade offs between them. This is due to the communication delays between nodes, a ubiquitous issue in blockchains that are a cause of race conditions and provide leverage to adversarial entities; both causing *forks*, a situation where more than one block extends the same block, causing confusion as to which one to extend. Forks are necessary to avoid as the adversary can use them to violate the integrity of the data records. In addition to the violation of data integrity, the adversary can also attempt to stall the chain or dominate it (only adversary generated blocks added to the chain).

Some security requirements, such as the one mentioned above, have to met for a blockchain to be secure enough to be used at large, popular blockchains are typically valuable and get attacked, others not so much. A blockchain can have some optional features like true *anonymity*, *recovery protocol*, etc. Typically, given the security that the blockchain is aiming for, the performance is optimized over. The number of transactions per second and the time it takes to confirm a blocks addition to the blockchain are the primary aspects of a blockchains performance. They are the core specs of a blockchain protocol that underlies everything built on top of it.

2.11 Blockchain Evolution

Bitcoin introduced blockchain technology. It established blockchains as a practical solution to distributed trust. It presented simple yet comprehensive mechanisms that served specific purposes. As far as the ecosystem is concerned, Bitcoin, addressed it in two ways: incentive mechanisms and inflation control. The incentive mechanism were monetary rewards for the miners to participate in mining. The incentives were of two types *block reward*, given to the miner that publishes his block to the blockchain, and the *transaction fee* which was fee given to the miners by the users as incentive to prioritize their transactions. Since block rewards involved adding currency to the system, risking unbounded *inflation*, Bitcoin, halves block rewards every four years to achieve a maximum currency cap of $21million$ BTC. For the chain construction, Bitcoin uses *Proof-of-Work* (PoW) mechanism and *longest chain rule*. PoW mechanism is used to essentially have a random miner add a block to the blockchain every 10 minutes (on average). The time gap of 10 minutes is to ensure that the network hears about the block before the next one is published to prevent collisions. The miner is selected with probability proportional to their *relative mining power*, which is their computational power as a fraction of the total computational power in the network. The PoW mechanism is executed as such: The miner must form a block, with its header and set of transactions (data), it must find the hash of this block and compare to a value called the *difficulty value*, if the hash is lower than it then the block is valid and can be broadcasted to the network to be added to the chain. The miner must repeatedly do this until he finds such a block. The difficulty value is adjusted, every two weeks, on the recent history of block timings, to ensure that a block is produced every 10 minutes. The longest chain rule enabled miners to select the chain to extend on the basis of their length. This rule is important to have as practically there would be some (natural or adversarial) collisions that would lead to forking (one block extended by two blocks). As nifty as a solution as this is, it is quite a rudimentary one. Various pieces of Bitcoin interact in conflicting ways: the inflation control mechanism threaten to rid the miners of their primary source of income rendering the Bitcoin transaction fee market too competitive; there is an inherent trade-off between the security and performance, one can not adjust the parameter to improve one without adversely affecting the other.

As research on Bitcoin ecosystem progressed, many issues came to light regarding it. And while research continues, these issues are largely unsolved. Bitcoin ecosystem problem persist in other blockchain too. Most blockchains still do not directly address *fairness* and the *health of the blockchains* until the situation is dire. Some ignore it entirely (in favor of a free market or easy development) and perhaps would eventually devolve into chaos. This apprehension to deal with such an important part of the blockchain mainly is due to the fact that no widely applicable solutions exist and investing into a highly complex analysis for a particular use-case with no certainty of result is typically not financially beneficial. Blockchains that center their approach around maintaining a well-adjusted ecosystem, are likely the ones that will last the test of time and perhaps set the tone for other blockchains to follow.

Performance is key to the adopt-ability and usability of blockchains. There have been attempts to modify the Bitcoin protocol for better performance. However, the poor efficiency remained. So different mechanisms altogether were tried such as *Proof-of-Space*, *Proof-of-Burn*, *Proof-of-Elapsed-*

Time, etc. The most popular of these alternative mechanisms is *Proof-of-Stake* (PoS) [31, 41], used by the Ouroboros protocols. PoS mechanisms are highly efficient and performant and are capable of being very secure if designed correctly. PoS mechanisms primarily use time periods and *verifiable random functions* (VRFs) as a replacement for PoW. PoS selects miners stochastically in each time period with probability proportional to their *relative stake* (currency held in the system), which is easily derived from the database. The selected miner(s) gets an opportunity to publish a block in that time period. This stochastic selection is done using VRFs with an input *random seed*. If the VRF output can be utilized in various ways to determine the selected miner(s). This PoS mechanism can then be coupled with a chain selection rule, to resolve forks when they are realized. While PoS blockchains avoids the efficiency issues of PoW, it is still subject to a similar trade-off between performance and security, and hence it too faces immense challenges when attempting to increase performance while maintaining security.

We have discussed a high level overview of preliminary blockchain concepts. There are many other attempts to scale. The interested readers are referred to [2, 14, 16] and the references cited therein. Over the next two chapters, we present our works, providing further details into the relevant concepts.

Chapter 3

Fairness and Health of the Ecosystem

3.1 Introduction

In blockchain technology, introduced in Bitcoin [55] by Nakamoto, the *miners* validate transactions that, the *users publish* (create and broadcast). Miners add valid transactions into the next block(s). Different miners attempt to publish (create and broadcast) the next block. In *Proof of Work* (PoW) blockchains, such as *Bitcoin*, the miner who solves a cryptographic puzzle first is whose published block is accepted as the extension. Each miner has a different puzzle, yet of the same level of difficulty, which needs computations to solve.

In Bitcoin, there are two types of rewards offered to the miners: *block rewards* and *transaction fees*. Block rewards are incentives that the miners are allowed to pay to themselves, minting currency in every block mined, regardless of the contents of the block. Transaction fees, on the other hand, are incentives offered by the users to the miners to prioritize their transactions. In Bitcoin and similar PoW blockchains, the miners invest resources, such as electricity and hardware, in such computations in anticipation of these rewards.

It is due to the investment on the part of the miners that they benefit from the *health of the blockchain*. In the context of this chapter, we characterize the *health of a blockchain* by (i) the fraction of mining power held by honest nodes; higher, the better (ii) *processing latency*: one of the performance parameters of the blockchain; lower the better, and (iii) the *price of consumption*; lower the variance, the better. All of these three metrics are linked to the perceived value of the blockchain and its currency. If the health of the blockchain is good, it is prudent to say that the underlying crypto-currency possesses a good value.

One of the key differences between traditional currency and *Bitcoin* is inflation control. To control inflation, block rewards are halved every four years in *Bitcoin*. Over time, a scenario develops, in which block rewards are negligible, and the only incentive for the miners is the transaction fee, i.e., the *transaction-fee-only model* (TFOM).

When the block rewards are high in value, we say the *Bitcoin* protocol satisfies *individual fairness for the miners* as it is believed that the current block rewards at least cover the marginal costs of mining

blocks. Since the miners are not hard-pressed for revenue, they can include transactions for free in the order that they arrive. This not only ensures that no transaction is stranded but also ensures that the price of consumption does not rise. A blockchain ecosystem is *fair for the users* if it has (i) low processing latency and (ii) stable price of consumption. Currently, the Bitcoin ecosystem is fair to the miners and users. The vital question we study is, do these three notions of fairness carry forward to TFOM?

The authors in [12] showed that in TFOM under low *influx* (incoming volume of transactions), the rational miners will *undercut* instead of following default strategy. While this analysis considers the impact of rational miners in TFOM w.r.t. *forking*, it does not consider the processing latency and the price of consumption.

In this chapter, our goal is to quantify fairness to the miners and the users and study the impact of TFOM under *standard influx*. Standard influx refers to the case when influx on an average is equal to the maximum outflux (processing capacity) of the blockchain. The two conditions, TFOM and standard influx, inherently go hand-in-hand as the Bitcoin matures [9]. Thus, making it very important to study and contemplate such scenarios. In this chapter, we analyze Bitcoin in TFOM and under standard influx, which we term as *mature operating conditions* (MOC), and show that it is unfair for both miners and users. This unfairness, in a nutshell, is exemplified in Fig. 3.1. We observe that those paying lesser transaction fees are expected to wait upto 9 blocks, and those who pay an insignificant amount of fees are expected to experience a processing latency of 14 blocks.

In TFOM, even when transaction volumes are sufficiently high enough to fill the processing capacity of the blockchain, it is not assured that the miners earn sufficient revenue. Insufficient transaction fees can be a major issue, as this puts the blockchain at a security risk due to the possibility of reduced honest mining power, which in turn will deteriorate the health of the blockchain, reducing its value, and hence further dropping of the honest miners.

Miners following *First-In-First-Out* (FIFO) processing ensures low and reasonable processing latency, avoiding stranding transactions entirely. Since no transactions are stranded, the price of consumption does not increase. Hence, miners following FIFO help sustain the blockchain's health through good performance and maintaining a stable price of consumption. In Bitcoin, under MOC, we show that miners following FIFO processing take heavy losses as compared to the ones mining greedily. This is an issue, as miners depend entirely on transaction fees to sustain mining and cannot take considerable losses in order to follow FIFO processing. This results in all miners processing transactions greedily. As we show in our analysis, this causes transactions to experience unreasonably high processing latency; such transactions are referred to as *stranded transactions*. This leaves the users with uncertainty about whether or not their transactions will be processed. A compounding effect of stranded transactions is the rising price of consumption. These issues culminate in unfairness for both the miners and the users. Thus, we say that Bitcoin, in its current form, is unfair under MOC (Proposition 1).

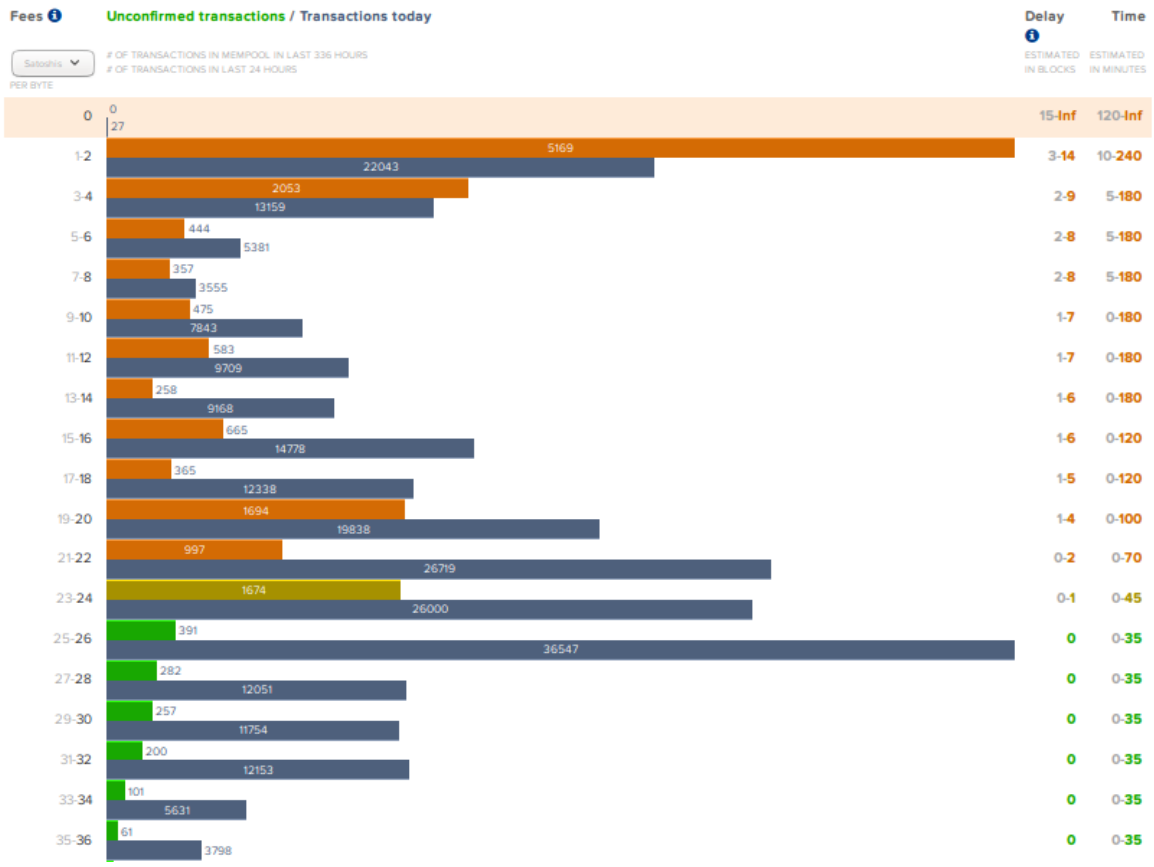


Figure 3.1: Transaction Fees vs Processing Latency (bitcoinfees.earn.com [25])

We solve these issues of unfairness by proposing a novel protocol, *BitcoinF*[1], for processing transactions. *BitcoinF* enforces a minimum transaction fee and uses two queues, instead of one to process transactions. *BitcoinF* allows the users to express urgency and have their transactions prioritized, just as they can do in Bitcoin. Our game-theoretic analysis proves that the proposed modification to Bitcoin, *BitcoinF*, ensures good health of the blockchain. Thus, we believe *BitcoinF* will lead to a stable ecosystem and hence will be fair to the miners and the users (Proposition 2). While there have been many published works analyzing TFOM, using collected data or using game-theoretic models, to the best of our knowledge, this is the first formal attempt at solving the pressing issues that are bound to arise in TFOM.

3.2 Related work

There are many papers in the literature studying transaction fees offered in *Bitcoin*. The authors in [53] study the behavior of transaction fees over a period of time, whereas Li et al. [50] conduct a theoretical analysis of a queuing game to study the transaction fees, and both conclude that the users paying lesser fee faced higher processing latency. The authors in [32], point out that if transaction fees were to be determined by the free market alone without a block size limit, it would be detrimental to *Bitcoin* as the fees would eventually become zero and miners will no longer have an incentive to mine. Easley et al. [26] develop a game-theoretic model, based on observational data, to study transaction fees and explain the behavior of miners and users in equilibrium. The authors in [45] conclude briefly that transaction fees would not play any major role unless the underlying rules of *Bitcoin* are changed. However, [33] suggests otherwise, clearly stating the importance of transaction fees and suggest increasing block rate to check congestion and increase miners' revenue. The authors in [38] study the effects of transactions paying a small fee and block size limit on the transaction confirmation time using queuing theory. The authors in [44] study how the users tend to offer high fees for their transactions to get included in the block when the demand exceeds block capacity, they model the confirmation time as a particular stochastic fluid queuing process.

Note that in follow-up work, [35] showed that block rewards are necessary. However, their model does not assume MOC. Under MOC, those block rewards could be modelled as our FIFO queue.

3.3 Preliminaries

In this chapter, we focus on Bitcoin when operating under *mature operating conditions* (MOC), i.e., when blocks rewards are negligible (Transaction-fee-only model (TFOM)), and there is standard demand (influx on an average is equal maximum outflux). First, we define all the important terms.

[1] A preliminary version of this work appeared in AAMAS 2020: S. Siddiqui, G. Vanahalli, and S. Gujar. Bitcoinf: Achieving fairness for bitcoin in transaction fee only model. In *International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2020*, 2020

Then, we present our model and describe our assumptions. Next, we explain how we simulate miners' behavior and users' behavior.

Important Definition

Definition 1 (Honest miner)**.** *We say a miner is* honest *or* non-adversarial *if it does not willingly attempting to disrupt the Bitcoin ecosystem by adding* invalid transactions *to blocks, attempting to* double spend *or by extending other than the* longest chain*.*

Definition 2 (Rational Miner)**.** *We say a miner is* rational*, if it continues mining when* individual fairness for the miners *is guaranteed and acts in favor of the health of the blockchain if the cost of doing so is marginal.*

The act of *processing transactions* simply involves selecting the transactions from the set of received but yet unprocessed transactions, adding them to the block and then publishing the block. This is also known as mining, and it is performed by miners. The system requires a *honest majority* of miners to maintain the security of the blockchain, vis-a-vis *persistence*, against adversarial miners. Persistence is a property that must be ensured by blockchains; It ensures that the *confirmation* (different from processing transactions) of a transaction by an honest node is never disputed by any other honest node. In this chapter, we consider that all miners are honest but *rational*. Miners are incentivized to participate in honest mining by rewards. If the miners are not compensated appropriately for their mining efforts, the rational, though honest miners may choose not to mine. Thus reducing the honest power in the network, weakening the blockchain against adversarial attacks, adversely affecting the health of the blockchain. When a miner chooses to stop mining, they are essentially giving up on the value of the blockchain and hence giving up on the significantly high investment they have in it, either in the form of the mining equipment or in the form of the blockchain currency token they hold.

Besides persistence, another property that must be ensured by blockchains is *liveness*. Liveness ensures that a transaction will eventually be processed; however, it is not sufficient as it does not guarantee that transactions will not get *stranded* for a long time, let alone be processed in a reasonable amount of time. As the blockchain technology matures, to maintain competitive performance, a blockchain must ensure that transactions are processed within a reasonable amount of time, i.e., low *processing latency*. Furthermore, another reason to avoid *stranded transactions* is that it leads to increasing *price of consumption*, further deteriorating the *health of the blockchain*. While stranded transactions can be avoided by simply restricting the amount of transaction fee offered to a single value, this trivial solution is unacceptable as the users' ability to offer a range of fees is required to express urgency and importance in a setting where there is varying demand.

Definition 3 (Processing Latency)**.** Processing latency *refers to the duration, which we measure in terms of blocks, between a user publishing a transaction and a miner processing it (publishing a block containing it).*

Definition 4 (Stranded Transactions)**.** Stranded transactions *are those transactions, that experience unreasonably high processing latency (* > 100 *blocks).*

Definition 5 (Price of Consumption)**.** *The price of consumption is the minimum transaction fee, as perceived by the users, that must be paid for the transaction to be processed.*

Definition 6 (Health of a Blockchain)**.** Health of a blockchain *is characterized by (i) the fraction of mining power held by honest nodes (ii)* processing latency *and (iii) the* price of consumption*.*

For the security of the blockchain, mainly Bitcoin, more than 50% of miners should be honest; higher, the better. This is an essential aspect for the system to be fair to both the honest miners and the users, as the decentralized nature of the system depends on it. Processing latency, one of the performance parameters of the blockchain, is the time taken to process a transaction. It is crucial to the blockchain's usability and adopt-ability. It also ensures that transactions are not *stranded* in the system and are added to the blockchain within a reasonable time. The price of consumption is the minimum transaction fee that must be paid for the transaction to be processed; for stability; lower the variance, the better.

The health of the blockchain would be better if more miners are honest. Given the miner's stake in the ecosystem they participate in, it is fair to say that they would act in favor of the health of the blockchain as long as the cost of doing so is marginal.

Definition 7 (Individual Fairness for The Miners)**.** *We say the given blockchain protocol satisfies* individual fairness for the miners *if the rewards from a block are at-least the cost of mining it.*

Definition 8 (Fairness of The Users)**.** *We say the given blockchain protocol satisfies* fairness for the users *if the users experience*
(i) reasonable processing latency,
(ii) stable price of consumption (i.e., f_{min}*), and*
(iii) decreasing average processing latency with increasing η*.*

Definition 9 (Fair Blockchain)**.** *We say that the blockchain ecosystem is fair if it satisfies individual fairness for the miners and fairness for the users.*

Since acting in the blockchain's favor yields optimal results required for the blockchain to perform competitively, it is imperative that miners do not deviate from it. Deviations from FIFO processing

can not be clearly detected and hence can not be actively discouraged; the only solution is to create an environment that ensures that the miner cannot benefit from such deviations. We quantify this as a game-theoretic equilibrium. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_k\}$ be the set of miners and S be the set of strategies for the miners in the blockchain to act upon.

Definition 10 (ϵ-*Expected Dominant Strategy Equilibrium*)**.** *We say $s = (s^*_{m_1}, s^*_{m_2}, \ldots, s^*_{m_k}), s^*_{m_i} \in S$ is ϵ-Expected Dominant Strategy Equilibrium for the miners if for all the miners,*

$$\mathbb{E}f_{txn}(s'_{m_i}, s_{-m_i}) < (1+\epsilon)\mathbb{E}f_{txn}(s^*_{m_i}, s_{-m_i})\ \forall s_{-m_i} \in S_{-m_i},\ \forall m_i \in \mathcal{M}$$

where $s'_{m_i} \neq s^*_{m_i}$. s_{-m_i} and S_{-m_i}, indicate the strategy profile and set of strategy profiles, followed by the miners except m_i. The expectation is w.r.t. randomness in influx of the transactions and the variance in the transaction fees.

The above definition may be too strong and difficult to achieve. Hence, we also work with the following, a weaker equilibrium concept from game theory.

Definition 11 (ϵ-*Expected Nash Equilibrium*)**.** *We say $s^* = (s^*_{m_1}, s^*_{m_2}, \ldots, s^*_{m_k})$, $s^*_{m_i} \in S$, is ϵ-Expected Nash Equilibrium for the miners, if for each miner, the expected revenue per block by following any strategy is not more than $(1+\epsilon)$ times what it would have obtained by following $s^*_{m_i}$; provided the other miners are following $s^*_{-m_i}$. I.e.,*

$$\mathbb{E}f_{txn}(s'_{m_i}, s^*_{-m_i}) \leq (1+\epsilon)\mathbb{E}f_{txn}((s^*_{m_i}, s^*_{-m_i}))\ \forall s'_{m_i} \in S,\ \forall m_i \in \mathcal{M}$$

where, $s^*_{-m_i}$ indicates the strategy profile in s^* by the miners except m_i. The expectation is w.r.t. randomness in influx of the transactions and the variance in the transaction fees.

The notation and acronyms used in this chapter are summarized in Table 6.1.

3.4 Model

For a tractable analysis, we simulate the execution of the Bitcoin protocol in steps of 10 mins, i.e., each block is published (created and broadcast) every 10 mins. Typically, the time duration between two consecutive blocks is random, with the expected value of 10 mins. We assume that each block can contain a maximum bs_{max} number of transactions. It is important to note that the maximum size of a block in *Bitcoin* is an inherent limitation of the Bitcoin protocol [57, 51]. In our analysis, we assume, in accordance with the standard influx condition under MOC, the number of transactions that arrive in each step is *not* constant and follow a Poisson distribution with mean bs_{max}. A Poisson distribution is

prudent here as the influx is a discrete number of events (w.r.t. arriving transactions) occurring in step, which is a fixed interval of time, with a known constant rate of bs_{max}. Hence, the average influx (in terms of the number of transactions) is equal to the maximum outflux (i.e., maximum block size). Here, *outflux* is the transactions that are processed by the miners.

We split the transaction fee offered by the user as, $f_{txn} = f_{min} + f_{extra}$. f_{min} is the minimum amount of fee, as perceived by the user, that must be included for the miner to process the transaction; it reflects the price of consumption. $f^0_{min} \leq f_{min}$, is the minimum transaction fee set by the protocol, and hence it is the initial value of f_{min} that the users start with. As there is no restriction on the minimum f_{txn} in Bitcoin, f^0_{min} is 0 as per the protocol. The f_{extra} is the extra fee the user would like to give to the miners to prioritize their transaction over others.

We model the aggression of a user towards f_{extra} through a parameter $0 \leq \eta < \infty$; higher the η, higher the f_{extra}. Each user, when publishing the transaction, calculates the amount of extra fee they would like to pay as a monotonically increasing function, $f_{extra} = \phi(\eta)$. Let $\psi_\lambda(\eta)$, a *pdf* characterized by a static parameter λ, be the distribution that captures the fraction of users having aggression level towards f_{extra} as η.

During each step, transactions are collected by the miners. At the end of each step, the miners form a block out of the currently unprocessed transactions followed by "instantly" publishing it. This merely a simplification of the process of continually updating the block while attempting to solve the cryptographic puzzle along with the assumption that a block would be mined by the end of the step.

We consider two modes that the miners may operate in; i) *greedy* processing, where miners greedily include transactions, i.e., they include the highest *or* lowest valued transactions, whichever may be more profitable ii) FIFO processing, where miners process transactions on a *First-In-First-Out* basis. These two modes of miner operation form the strategy space of the miners. Since the influx, as well as outflux, is in terms of the number of transactions, it is understood that all transactions are considered to be of the same size in terms of the space taken on the block. Hence *greedy* processing only considers the transaction fee offered and not the size of the transaction in bytes.

Assumptions on Miners' Behaviour

We assume that all miners are honest but rational. The rationality of the miners implies the following:

- Miners would continue to mine and sustain the blockchain, as long as mining costs are covered.
- Since the health of the blockchain is crucial to the value miners obtain from mining, and that all miners inherently understand this, miners act in favor of sustaining the health of the blockchain, i.e., follow FIFO processing, if the cost of doing so is marginal.

Assumptions on Users' Behaviour

We assume that the user would like to have his transaction processed within a reasonable time and that their inclination to have their transaction prioritized is characterized by their aggression level, η towards paying a higher f_{extra}. This assumption implies the following:

- The user chooses f_{txn} by first considering the f_{min} and then deciding $f_{extra} = \phi(\eta)$ where η captures its aggression parameter. Note that, the system need not know the η for each individual user, but for analysis, we use the distribution of users against η (i.e., $\psi_\lambda(\eta)$ is known).

- The user observes transactions, below a certain threshold of fees, being stranded, they concede to making the presumption that this threshold of fees is the new f_{min}, in the sense that transaction below this fees will not be processed in a reasonable time. This is because a user will not attempt to publish a transaction if they do not expect it to be processed.

3.4.1 Simulation Setup

Since the theoretical analysis is intractable in such a complex scenario. We use simulations to support our arguments. For a fair and consistent analysis, we use the same simulation setup and parameters to simulate both Bitcoin as well as our protocol. All simulation results were averaged over 10 runs.

As mentioned in Section 3.4, the execution of the protocol in consideration, proceeds in steps. Each step represents the time between blocks. At the end of each step, a block is added to the chain. Each block has a maximum capacity, $bs_{max} = 1000$, in terms of transactions. All the simulations are run in the setting where the average influx is Poisson distributed with mean equal to bs_{max}.

We take, $\psi_\lambda(\eta) = \lambda \cdot e^{-\lambda \cdot \eta}$, an *exponential distribution* characterized by $\lambda = 3$, where λ is the rate parameter of the exponential distribution. ϕ, the function used to calculate the f_{extra} a user with aggression parameter η gives, we take to be $\phi(\eta) = f_{extra} = e^\eta - 1$.

We take the granularity of the size of miners in terms of their mining power to be $\delta = 0.05$; and $0 \leq \beta \leq 1$ to be the fraction of miners that follow *greedy*, while the rest follow FIFO.

To observe the cost of mining as per FIFO as opposed to acting *greedy*, and further investigate the establishment of equilibrium, we simulate both, *BitcoinF* and *Bitcoin* with varying values of β. We then consider the resulting *average revenue per block* mined for both FIFO and *greedy* behavior, in both *BitcoinF* and *Bitcoin*. The fraction of mining power controlled is represented in the simulation by setting the same fraction as the probability of mining a block.

To emulate the user's characteristic of observing the change in f_{min} based on the stranded transactions, we use epochs of observation. Each *observational epoch* is a series of subsequent steps at the end of which the users change their f_{min} accordingly. At the end of each epoch, the users check the average processing latency of transactions that were published in this epoch. The highest transaction fee that, experiences an average processing latency high enough to be considered stranded, is considered to be the new f_{min}. The length of the observational epoch, i.e., the number of steps the users observe after

which they change their presumption of f_{min}, we take to be 1000 steps. We use 100 to be limit to the processing latency in units of steps, after which we consider a transaction to be stranded.

Now we study *Bitcoin* protocol in the next section.

3.5 Bitcoin Transaction Processing: Analysis under MOC

First, we describe *Bitcoin* protocol, and then explain what assumptions we make, highlight the specifics of *Bitcoin* simulation, and the inference from the simulations.

3.5.1 Bitcoin Protocol

In Bitcoin, the market for "space on the block" is a completely free market (FM), there is no regulation on how the transactions must be processed or how much transaction fee must be given. In Bitcoin, a block contains only one section (we refer to this as the FM section) where there are no restrictions (except our assumption on the maximum number of transactions per block, i.e., block size). The users publish only one instance per transaction containing $f_{txn} = f_{min} + f_{extra}$. Initially, $f_{min} = 0$ as the Bitcoin protocol does not state any minimum transaction fee that must be included.

In TFOM, each miner must collect transaction fees to sustain their mining efforts. Even when assuming an influx of transactions that is on an average sufficient to fill the blocks, there is no guarantee that the incoming transactions will contain sufficient transaction fees to sustain the mining efforts. Hence, individual fairness for the miners can not be established.

In Bitcoin, while the block rewards alone are sufficient to sustain mining efforts, the rational miners can be expected to process transactions in a FIFO manner, especially since the number of users offering a competitive fee is minimal. However, under MOC, we assume in our analysis that when processing transactions from the FM queue to be added to the FM section of the block, the miners pick the transactions offering highest transaction fees, as shown in Fig. 3.2a. The miners could choose to follow FIFO while processing transactions from the FM queue. They might want to do this to preserve the health of the blockchain. However, we expect our simulation-based game-theoretic investigation into the establishment of equilibrium will yield that the miners lose a large part of their revenue by following FIFO processing in the FM queue. Hence the miners cannot be expected to act in favor of the blockchain's health as the loss is considerable. Thus, we assume the miners will gather transactions greedily.

The users suffer as a consequence of the miners not being able to follow FIFO processing. Not wanting to follow FIFO would not be cause for concern to the users if the influx of transactions to be processed was low. If the influx of transactions is low, then no matter the transaction fees, all transactions would be included in the next block. In low influx scenarios, the competition is not high, and the users realize that there is no need to pay any more than a marginal fee, as their transaction would get included in the next block regardless. However, as we show in our analysis, issues arise in higher influx scenarios where the users *must* pay competitive fees to have their transactions processed or risk having them

stranded. Further, since these stranded transactions are public knowledge, this, as we see in our analysis, causes the price of consumption to rise, adversely affecting the health of the blockchain.

In Bitcoin, under MOC, as we show using simulations, there are a large number of transactions that get stranded. This causes the users to be uncertain about when their transaction will be processed, or even if it ever will be. These stranded transactions are public knowledge. Thus, over a sufficiently long series of consecutive steps, if it is observed that there are transactions that are stranded, the users are likely to treat the highest transaction fees offered by the stranded transactions as the new f_{min} to avoid their transactions being pushed further down in the queue. Given this new f_{min}, the phenomenon will repeat. Depending on the observations of stranded transactions, f_{min} can increase, decrease (not below f^0_{min}), or remain constant. We study what is likely happen for f_{min} *Bitcoin* via simulations.

3.5.2 Simulation Specifics

Since the miners only include transactions greedily, to emulate miners in this scenario, we have only one section (FM section) of the block, which spans the entire capacity of the block, to which we keep adding the highest valued transaction till the block is full. As per the Bitcoin protocol, we keep the initial $f_{min} = 0$.

First, we conduct simulations to investigate the establishment of equilibrium, which we expect to state that to follow *greedy* processing is *ϵ-Expected Dominant Strategy Equilibrium* validating the assumption that the miners follow *greedy* processing from the FM queue. We assume the same in our simulation estimating average processing latency and the change in f_{min} with observation epochs.

3.5.3 Simulation Results and Inference

First, we discuss the result of our investigation of equilibrium of miner behavior in the *Bitcoin* ecosystem. The strategy space of the miners here is $S = \{FIFO, \textit{greedy}\}$, where $s \in S$ is the mode of processing (as in Section 3.4) of the miner in the FM queue.

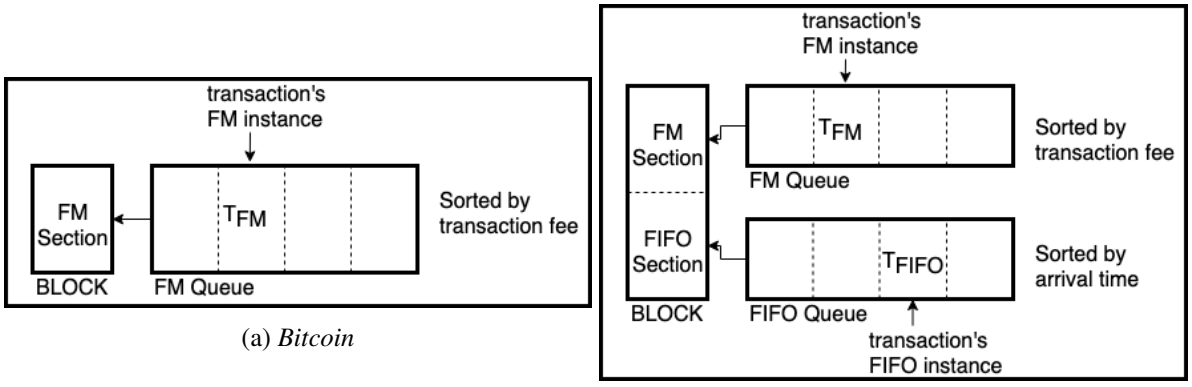
Clearly, from Fig. 3.3a, we see that:

$$\mathbb{E}f_{txn}(s'_{m_i}, s_{-m_i}) < (1+\epsilon)\mathbb{E}f_{txn}(s^*_{m_i}, s_{-m_i})\ \forall s_{-m_i} \in S_{-m_i},\ \forall m_i \in \mathcal{M}$$

where $s'_{m_i} = FIFO$, $s^*_{m_i} =$ *greedy* and $\epsilon = 0$.

Intuitively, miners make significantly higher revenue if they followed *greedy* processing as opposed to FIFO processing regardless of β. Thus we say that following *greedy* processing in the FM queue is *ϵ-Expected Dominant Strategy Equilibrium* with $\epsilon = 0$.

Confirming our assumption about miner behavior, we now discuss the results of our simulations regarding the fairness of the blockchain. As we can see from Fig. 3.3b, the f_{min} rises with observational epochs. This causes instability in the price of consumption. The figure also implies that when the influx has not been *standard* for long enough, Bitcoin cannot ensure that mining costs will be covered, i.e., individual fairness for miners is not guaranteed. In Bitcoin, Fig. 3.3c shows that, the users that have very



(a) *Bitcoin*

(b) *BitcoinF*

Figure 3.2: Transaction processing in *Bitcoin* and *BitcoinF*

little aggression towards paying f_{extra}, experience unreasonably high processing latency, i.e., stranded transactions. These stranded transactions, in turn, cause the price of consumption to rise, as seen in Fig. 3.3b.

These observations are summarized as Proposition 1.

Proposition 1. *If the miners strategy space is* $S = \{FIFO, \text{greedy}\}$*, it is* ϵ-Expected Dominant Strategy Equilibrium *for the miners to follow* greedy *with* $\epsilon = 0$*. As a consequence, the* Bitcoin *ecosystem is not a fair under MOC.*

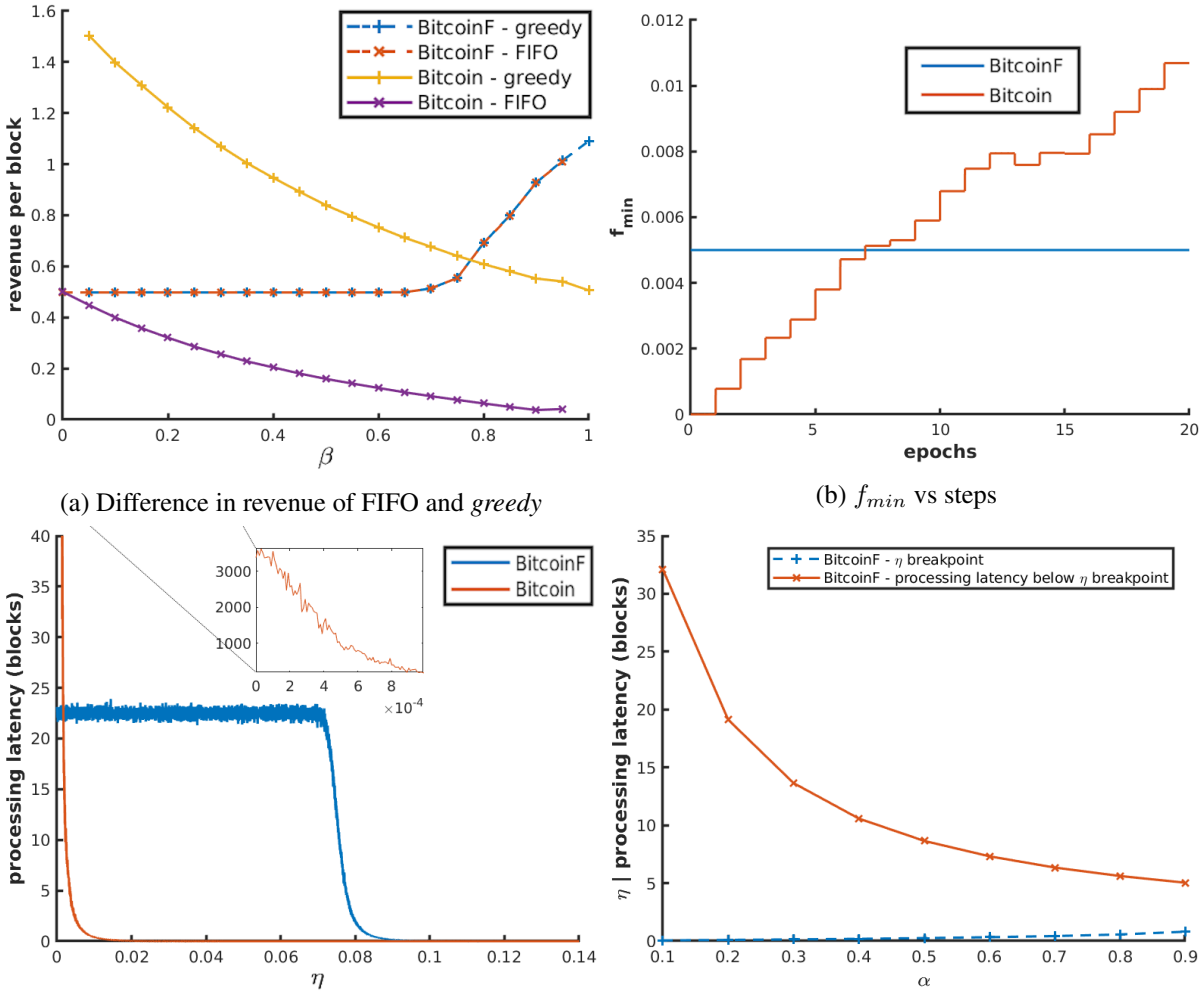
3.6 Achieving Fairness under MOC

As discussed in the previous section, under MOC, *Bitcoin* faces challenges. To resolve this, we propose a simple modification to *Bitcoin*, which we call *BitcoinF*.

3.6.1 *BitcoinF*

To solve the issues of unfairness for both the miners and the users, we propose a protocol to process transactions. Our approach is two-fold: Firstly, we enforce a minimum fee of $f^0_{min}(> 0)$ that is to be included in every transaction. Secondly, we introduce a section in the block that only accepts transaction instances with $f_{txn} = f^0_{min}$, called the FIFO section of the block. So now, there are two sections in the block: FM and FIFO. The FIFO section has a size of $\alpha \cdot bs_{max}$, whereas FM has the remaining. This is illustrated in Fig. 3.2b. In our simulation of *BitcoinF*, we set $\alpha = 0.2$.

Formally, we propose *BitcoinF* as a *block validation rule*. This rule will have two parameters, α, and f^0_{min}. Miners shall only accept and extend blocks that follow the rule. We expect that when this protocol is implemented, it will be enforced by the honest miners, as is commonplace in blockchain ecosystems.



(a) Difference in revenue of FIFO and *greedy*

(b) f_{min} vs steps

(c) Processing latency vs η

(d) Trend w.r.t. alpha

Figure 3.3: Simulation Results

Definition 12 (*BitcoinF*: Block Validation Rule)**.** *Each block must contain* $\alpha \cdot bs_{max}$ *transactions offering* $f_{txn} = f^0_{min}$.

A typical execution is described as follows:

- Users when they want to add a transaction to the blockchain broadcast two instances of the same transaction; one instance that has $f_{extra} = 0$, and the other instance where f_{extra} is as chosen by the user. Both instances must include at least f^0_{min} as required.
- The miners collect these instances of every transaction and add the instance with $f_{extra} = 0$ to the FIFO queue and the other instance, the one with f_{extra} to the FM queue.
- When an instance of a transaction is processed, the other instance is invalidated.

- The block can have transactions in the FM section *only* if the FIFO section of the block is completely filled. The honest but rational miners naturally would first add as many transactions as possible to the FM section from the FM queue (they may do this however they please, but naturally they choose the ones with highest transaction fees), while keeping aside sufficient transactions to fill the FIFO section of the block. Then, the miners fill the FIFO section of the block with transactions selected from the FIFO queue in a FIFO manner.

When processing transactions from the FM queue to be added to the FM section of the block, the miners naturally pick the transactions offering the highest fees. Further, we assume that while processing transactions from the FIFO queue, the miners follow FIFO.

The miners could choose to follow *greedy* processing, i.e., add the minimum valued transactions instead of following FIFO processing to fill the FIFO section. They might want to do this, as FIFO processing might process some slightly higher valued transactions through the FIFO section of the block, voiding the f_{extra} it offers; thus, by following *greedy* processing, the miners process the least valued transactions through the FIFO section, while keeping the slightly higher valued transactions (which would otherwise get processed through FIFO) for later, to be processed through the FM section of the block. However, we expect our simulation-based game-theoretic investigation into the establishment of equilibrium will yield that the miners gain a negligible profit by following *greedy* processing as opposed to FIFO processing in the FIFO queue. Hence the miners can be expected to act in favor of the blockchain's health as the loss is insignificant. Thus, we assume the miners will follow FIFO processing in the FIFO queue.

3.6.2 Simulation Specifics

The size of the FIFO section is set to be $\alpha \cdot bs_{max} = 200$ transactions, to which transactions are added in a FIFO manner. The size of the FM section is set to be $(1 - \alpha) \cdot bs_{max} = 800$ transactions, to which transactions are added greedily. The FM queue is processed before the FIFO queue, as is the expected behavior of the miners. To emulate the random order of the transactions' arrival during a step, when processing from the FIFO queue, random transactions are picked from the set of transactions with the highest processing latency. As per the protocol, initially, the value of f^0_{min} is set appropriately to compensate miners, we, in our simulation, set it to be 0.005.

First, we conduct a simulation to investigate ϵ-*Expected Nash Equilibrium* which we expect to state that to follow the FIFO is ϵ-*Expected Nash Equilibrium* validating the assumption that the miners follow FIFO processing from the FIFO queue. We assume the same in our simulation estimating average processing latency and the change in f_{min} with observation epochs.

3.6.3 Simulation Results and Inference

First, we discuss the result of our investigation of equilibrium of miner behavior in the *BitcoinF* ecosystem. The strategy space of the miners here is $S = \{FIFO, greedy\}$, where $s \in S$ is the mode of processing (as in Section 3.4) of the miner in the FIFO queue.

Clearly, from Fig. 3.3a, we see that:

$$\mathbb{E}f_{txn}(s'_{m_i}, s^*_{-m_i}) \leq (1+\epsilon)\mathbb{E}f_{txn}((s^*_{m_i}, s^*_{-m_i}))\ \forall s'_{m_i} \in S,\ \forall m_i \in \mathcal{M}$$

where $s^* = \{FIFO, FIFO, \ldots, FIFO\}$ and $\epsilon = 0.00037$.

Intuitively, miners gain negligible profit if they followed *greedy* processing as opposed to FIFO processing in the FIFO queue when $\beta = 0$. Thus we say that all miners following FIFO processing in the FIFO queue is ϵ-*Expected Nash Equilibrium* with $\epsilon = 0.00037$.

Here, ϵ-*Expected Nash Equilibrium* is clearly a much weaker property than the one established by *BitcoinF*. Motivated by the analysis in [68], where the authors consider a fraction of agents (our case miners) following honest strategy and remaining agents follow greedy strategy, it is easy to see that our protocol exhibits ϵ-*Expected Dominant Strategy Equilibrium* with $\epsilon = 0.00037$ when $\beta \leq 0.55$. The protocol does not establish ϵ-*Expected Dominant Strategy Equilibrium* for all β, as after a point, the fraction of miners following FIFO processing in the FIFO queue drops low enough that transactions start to get stranded, raising the price of consumption and hence raising the average revenue of all miners.

Confirming our assumption about miner behavior, we can now discuss the results of our simulations regarding the fairness of the blockchain. As seen in Fig. 3.3b, f_{min} remains constant with time. This implies a stable price of consumption. Since the f^0_{min} can be set as per requirements to cover mining costs, *BitcoinF* guarantees individual fairness for miners under MOC. Fig. 3.3c shows that in *BitcoinF*, no matter the users' aggression towards paying f_{extra}, the users experience reasonable processing latency. Since there are no stranded transactions, the price of consumption does not rise, as seen in Fig. 3.3b. In fact, under MOC, the users would know what processing latency to expect as soon as they publish the transaction. If their η is higher than a certain η *break-point*, then their transaction will get processed almost immediately; if not, they know the upper bound on the processing latency they will experience. This trade-off between the η break-point and processing latency below the η break-point as a function of α is visualized in Fig. 3.3d. These simulation based observations can be summarized as Proposition 2.

Proposition 2. *If the miners strategy space is* $S = \{FIFO, \text{greedy}\}$*, it is* ϵ-Expected Nash Equilibrium *for the miners to follow FIFO with* $\epsilon = 37 * 10^{-5}$*. As a consequence,* BitcoinF *ecosystem is a fair under MOC.*

3.6.4 Security Analysis

A strategic and intelligent miner could attempt to use our protocol to leverage an unfair advantage. In this section, we show that such attempts are not quite effective and hence, do not threaten our protocol.

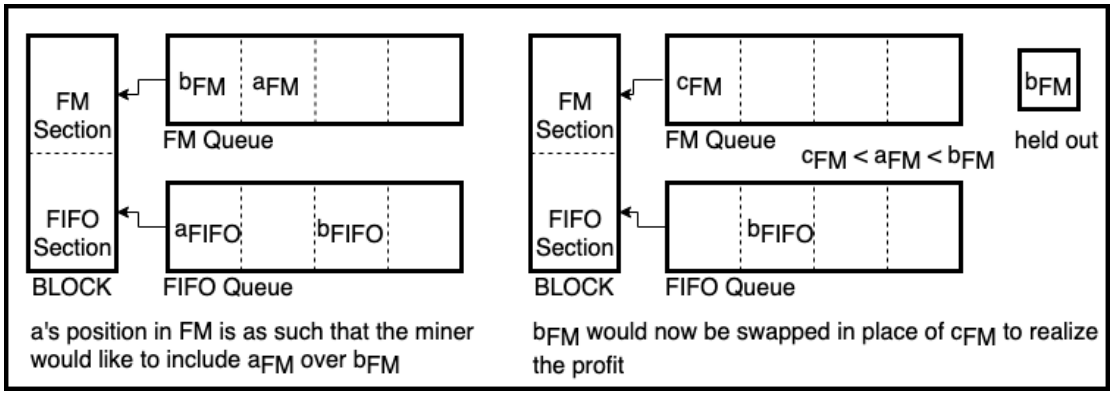


Figure 3.4: Depiction of swapping

Ignoring the FIFO Section of the block The miners would ideally like to ignore all the f^0_{min} instances of the transactions. The miners cannot do this as any valid block requires that the FIFO section of the block must be filled entirely before any transactions are added to the FM section of the block. Also, all transaction instances in the FIFO section are only supposed to offer f^0_{min} fee.

Swapping in transactions that are about to be processed through FIFO queue The miner can always process a transaction of lower value, say transaction a, than it should from the FM queue, by swapping out a transaction of higher value, say transaction b. Please refer to Fig. 3.4. The miner might be inclined to do this if a is about to be processed via the FIFO queue, whereas b is not. The miner might want to do this if he notices that a is, in fact, of higher value than what is typically included in the FM section. In this sense, the miners keep b held out, to be swapped in later (finally) for another transaction, say transaction c, that is lower in value as compared to b and a, thus realizing a profit. The miner may swap several times (swapping out transactions b, b', b'' ...) before finally swapping out transaction c.

Note that during this attempt, it is easy to see that the miner is risking losing a profit as he is betting on finding the transaction c. He may not find such a transaction if the lowest value of the transactions included in the FM section never drops sufficiently, or the miner does not maintain a mining monopoly, and some other miner mines the next block gaining the risked amount.

The theoretical analysis of this attack strategy is not tractable, and nor is the simulating the attack feasible due to the complexity of the actions available and state-space. Thus, to show that this attack is ineffective, we give the adversary generous and impractical advantages and show that even in the best case of executing this attack in the backdrop of our simulation, the adversary gains as little as less than 1% of the total rewards gained.

Since the transaction a must ultimately be swapped with a transaction c, we simply consider the number of potentially ultimately successful swaps, as the minimum of; the number of transactions processed by the FM queue, that are valued below the maximum value of the transactions processed by the FIFO queue; and the number of transactions processed by the FIFO queue, that are valued above the minimum value of the transactions processed by the FM queue. We multiply the value obtained by

the difference between; the maximum value of the transactions processed by the FIFO queue and the minimum value of the transactions processed by the FM queue. In our simulations, the resultant value turns out to be less than $0.59 \pm 0.29\%$ of the total value of transactions processed.

Now, this is clearly an over-valuation for the following reasons; (i) The risk of the attempt is completely ignored here. (ii) The number of ultimately successful swaps considered is the result of the best (perhaps better than the practical best) possible exploitation of the one-to-one correspondence of the swapped transactions by the adversary. (iii) The value of profit gained with each successful swap is just taken to be the maximum profit gained from the best possible swap. (iv) The attacking miner is assumed to have a monopoly over mining, i.e., the attacking miner is the only one mining and hence can carry out this attack unhindered, i.e., without the possibility of another miner publishing a block impeding the attacking efforts.

3.6.5 Discussion

The parameters of *BitcoinF*, α and f^0_{min} can be chosen by consensus and should be agreeable by both the miners and the users; we suggest $\alpha = 0.2$ and $f^0_{min} = \frac{\text{average cost of mining a block}}{bs_{max}}$. If $\alpha = 0$, then *BitcoinF* reduces to *Bitcoin*, whereas $\alpha = 1$ would be strictly FIFO. *Bitcoin*, as we have seen, is not fair. Strictly FIFO processing disables the ability of the users to express urgency, and the average processing latency will not decrease with increasing η, which is a requirement for fairness for the users. An f^0_{min} too low will discourage mining, an f^0_{min} too high will discourage usage of the blockchain.

While we have chosen specific functions and parameters for ϕ and ψ, we believe that any monotonically increasing function for ϕ and any monotonically decreasing function for ψ would yield similar yet scaled results.

3.7 Conclusion

In this chapter, we studied *Bitcoin* under *mature operating conditions* (MOC), i.e., in TFOM and standard influx. To study a given blockchain, we introduced notions of fairness (i) for the miners and the users, and (ii) the health of a blockchain. Under reasonable assumptions, we showed using simulations that miners act greedily in *Bitcoin*, as it is *ϵ-Expected Dominant Strategy Equilibrium* to do so, and as a consequence, *Bitcoin* ecosystem is not fair. To achieve fairness in *Bitcoin*, we propose *BitcoinF*, a simple yet powerful modification to *Bitcoin*. In *BitcoinF*; each transaction must include a minimum amount, to ensure that transaction fees cover marginal mining costs under MOC; and must have a minimum number of transactions offering the minimum specified amount. We showed using simulation analysis that in *BitcoinF*, miners act in favor of the health of the blockchain as it is *ϵ-Expected Nash Equilibrium*, and as a consequence, *BitcoinF* ecosystem is fair.

Chapter 4

Security and Performance

4.1 Introduction

In any blockchain system, delays in communication and adversarial attacks may cause *forks* in the chain, creating ambiguity as to extend which block. In summary, (i) to prevent forks, we require that only one *node* publish a block at a time. Thus, we need a *distributed mechanism*, which we refer to as *block publisher selection mechanism* (BPSM) to select a node as *selected block publisher* (SBP). (ii) Still, if forks occur, to resolve them, we require the protocol to specify a *chain selection rule* (CSR). The BPSM and CSR functionally characterize a blockchain protocol. Typically, we measure a blockchain's performance with the two metrics: i) *transactions per second* (tps), and ii) *time for finality* (t_f), i.e., the time required to confirm a transaction.

Bitcoin blockchain protocol uses a *Proof-of-Work* (PoW) based BPSM and *longest chain* as the CSR. In the PoW mechanism, each node has a different cryptographic puzzle to solve, of the same level of difficulty. Once a node solves the puzzle, it publishes the block, i.e., proposes the block to be added to the chain. The rest of the nodes verify and add the first valid block they receive to their blockchain. Block rewards and transaction fees serve as incentives for nodes to participate and follow the protocol. Bitcoin, though a great innovation, has certain challenges when it comes to efficiency and performance. Bitcoin consumes a high amount of electrical power to function. As of writing this chapter, the Bitcoin network uses an estimated 138×10^3 Peta hashes per second [6] and annual power consumption of 70 TWh [22]. Typically in Bitcoin $tps = 4$ and $t_f = 60$ minutes [8]. Due to the poor tps, it is unable to scale to high volumes, and the high t_f deters usability and adoption. Attempts such as *GHOST* [62] aim to improve the performance by using a different CSR. In GHOST, the BPSM being still PoW-based, its power consumption remains high. Towards addressing high power consumption, researchers proposed *Proof-of-Stake* (PoS) based BPSM.

PoS based blockchain protocols, such as Algorand [31] and Casper [10], stochastically select an SBP with probability proportional to its *relative stake*. This approach is also consistent with the expectation that nodes that are stakeholders would not want to endanger the system. The *Ouroboros* protocols

[41, 20, 3, 39] are popular PoS-based protocols, amongst others. Ouroboros is not only academically published[1] in a peer-reviewed forum, but also is the foundation for the crypto-currency *Cardano* [11].

Ouroboros v1 (v1) [41] achieves high tps and better t_f than Bitcoin. However, for a blockchain protocol to be completely secure, it must be immune to *fully adaptive corruption*s (FACs), i.e., a *dynamic adversary*. Ouroboros v1 is not immune to FACs. *Ouroboros Praos* (Praos) [20] uses a different BPSM. Praos does solve the security weakness, but does not improve performance. While both *Ouroboros Genesis* [3] and *Ouroboros Crypsinous* [39] protocols enhance the Ouroboros protocol, they too do not improve performance.

In this work, we propose a novel blockchain protocol, *QuickSync*[2], that is secure against FACs, and achieves slightly better tps, and improves on the t_f by a factor of 3, as compared to Ouroboros v1. Essentially, it quickly synchronizes (resolves) the forks that arise. To build *QuickSync*, we employ the framework of the Ouroboros protocol. *QuickSync* differs from v1 and Praos, in both the BPSM and the CSR. The key idea is, we propose a metric called *block power*, assigned to each block. With this, we compute *chain power* of every competing chain. The chain power of a chain is the sum of block powers of all the blocks of the chain. Using chain powers, we establish the *best chain*, the one with the highest value for this metric, that the node must build on from a given set of chains. It results in ensuring that all forks are trivially resolved, except for the ones generated by the adversary. Block power is a function of the output of the node's privately computed *Verifiable Random Function* (VRF) output based on the digital signature of the node, seed randomness, and the slot counter. The VRF output is not revealed until the block is published, enabling immunity against Fully Adaptive Corruptions. Block power is also a function of the relative stake that the node holds to enable the PoS aspect of the mechanism. A naive implementation of such a concept is vulnerable to Sybil attacks. To solve this, we present a *Sybil attack resistant function*, which we utilize for the block power metric, with the help of a technique called *histogram matching*.

As multiple nodes publish blocks at the same time, it may seem that there will be several forks in *QuickSync*, as is the case in the other PoS protocols such as Praos. The key novelty here is that we resolve these forks immediately using block power rather than relying on the network to eventually resolve them using the longest chain CSR. Thus, *QuickSync* is in sharp contrast to other PoS-based mechanisms that *do not differentiate* between the published blocks.

Researchers showed that any blockchain protocol satisfying the three properties: *common-prefix*, *chain-growth*, and *chain-quality* implements a robust transaction ledger [28, 40, 56]. We prove that *QuickSync* satisfies these three properties (Theorem 1) and ascertain that our protocol is immune to FACs (Proposition 4). We also examine our protocol for different attack strategies and show resistance to them. In summary, the PoS-based blockchain protocol, *QuickSync* fixes the security weakness of Ouroboros v1 while improving on performance and performs better in terms of tps and t_f by about an order of magnitude as compared to Bitcoin.

[1] Some PoS protocols are not academically published, e.g. Casper

[2] A preliminary version of this work is under review at ICDCS 2021: S. Siddiqui and S. Gujar. Quicksync: A quickly synchronizing pos-based blockchain protocol. *arXiv preprint arXiv:2005.03564*, 2020

The rest of the chapter is organized as follows. In Section 4.2, we summarize the related work in this domain. In Section 4.3, we explain the relevant preliminaries. In Section 4.5, we describe our protocol, *QuickSync*. In Section 4.7 we discuss the intuition behind *QuickSync*. Finally, we discuss how to set the parameters for *QuickSync* in Section 4.10 and conclude the chapter in Section 4.11. The detailed security analysis, analysis of attack strategies, and a comparative note are provided in the Appendix.

4.2 Related Work

Bitcoin Protocol Bitcoin's popularity has attracted much research on this subject. Analysis of the functioning of the Bitcoin protocol, in [28, 56], provides a fundamental understanding of how it implements a robust transaction ledger, while a study of Bitcoin from a game-theoretic perspective, such as in [49, 46], provides insight into the realistic operation. The vulnerabilities of the protocol are well explored in works such as [27, 12, 67, 1]. There have been several efforts to improve the Bitcoin protocol, such as the one discussed in [62]. There is also much research based on utilizing blockchain technology for different purposes, as discussed in [24, 64, 52].

PoS-based Protocols As pointed out in the Introduction, to maintain Bitcoin, the nodes consume a large amount of power due to PoW-based BPSM. To avoid this massive power consumption, researchers explored alternative modes of mining currencies. PoS has been one of the most popular such modes. There have been several approaches to PoS-based protocols apart from the Ouroboros protocol, such as; *PPcoin* [42], which uses coin age along with PoS and PoW; Ethereum's *Casper* protocol [10], that uses validators that lose their stake when they behave maliciously; *Snow White* protocol [5], that too uses epochs and is provably secure; *Algorand* protocol [31], that uses *Byzantine Fault Tolerant* mechanisms to achieve consensus and is also provably secure, though it requires $\frac{2}{3}$ majority of honest nodes.

Apart from independent protocols, there has been significant research on concepts that could potentially improve or augment PoS protocols. Ouroboros Crypsinous [39] already utilizes one such technology, *Zerocash* [58], to provide privacy. PoS protocols could also use *context-sensitive transactions* [47, 48] as discussed in [29] to do without moving checkpoints. Efforts such as *Scalable Bias-Resistant Distributed Randomness* [63], could aid in improving the randomness beacons used in PoS protocols. Concepts such as, sharding as demonstrated in *Omni-Ledger* [43], and *Proof-of-Stake Sidechains* [30], could potentially be applied to scale the throughput of blockchains.

4.3 Preliminaries

In this section, we discuss the preliminary notions and key concepts required to build *QuickSync*.

4.3.1 General Concepts and Notation

4.3.1.1 Blockchain

A blockchain is a singular sequence of blocks connected by hash links, with each block linked to the last, starting from the genesis block. The genesis block enlists the initialization specifics and parameters of the blockchain. A block is comprised of a *block header* and *block data*. The block header contains, amongst other things, the publisher's public key and the hash of the *merkle tree root* of the block data, as well as the hash of the previous block. The block data contains the transactions or any other data that is to be added to the record. *Block publisher* is the node that has built the block in consideration. We refer to the number of transactions per block as tpb. A chain of blocks, C referred to as chain, consists of an ordered set of blocks, where every block, $B^l; l > 0$ is immutably linked to the block, B^{l-1}, $\forall B^l \in C$; where l represents the ordinal number, and B^0 is the genesis block. The length of a chain, C, is denoted as $len(C)$, is the number of blocks in the chain excluding the genesis block. We say that the chain selected by the CSR has been *adopted* and is *held* by the node.

4.3.1.2 Blockchain Protocol

Typically a collection of independent, autonomous, distributed nodes are participants in maintaining a blockchain, and they need a blockchain protocol for the same. We start by describing the key concepts to build any blockchain protocol. Each node is identified by a public key pk and holds a master secret key msk. $\mathbb{S}$ is the set of all nodes, composed of $\mathbb{H}$, the set of all honest nodes (motivated by reward schemes) and, $\mathbb{A}$, the adversary.

Blockchain being a decentralized system, to propose a new block, we need to select the publisher for each block. We call the publisher *selected block publisher* (SBP). As multiple nodes from $\mathbb{S}$ would be interested in writing the next block, we need *block publisher selection mechanism* (BPSM) to select an SBP. Due to network delays and adversarial attacks, it is possible to have multiple versions of the chains, i.e., forks from the previously agreed state of the chain. While designing a blockchain protocol, there should be an implicit or explicit rule to resolve these forks. We refer to this as *chain selection rule* (CSR). In summary, to design a blockchain protocol, we need to specify BPSM and CSR. For example, in *Bitcoin*, the BPSM used is PoW (whoever solves the cryptographic puzzle first), and the CSR used is *longest chain*.

4.3.1.3 Proof-of-Stake Blockchain Protocols

In *Proof-of-Stake* (PoS) protocols, each node $n \in \mathbb{S}$ has influence proportional to the amount of *relative stake* it holds. This influence is expressed as the probability of being selected as an SBP. The fraction of the total stake held by the nodes in consideration is referred to as *relative stake*. r_h and r_a denotes the relative stake of the honest nodes and the adversary respectively ($r_h, r_a \geq 0, r_h + r_a = 1$).

The relative stake that is active and participates in the execution of the protocol at any given moment is represented as $r^{active} \geq 0$.

4.3.1.4 Forks

A *fork* in a blockchain is the case when two different blocks extend the same block. Forks happen when two or more honest nodes publish blocks in temporal vicinity, close enough not to have heard of the other's blocks (due to network delay). Forks cause multiple possible valid blocks that can be extended. The adversary could also attempt to create a fork, privately or publicly, to compromise the protocol intentionally. These forks enable an adversary to double spend. An essential part of blockchain protocols is to ensure that these forks are resolved with increasing (preferably exponential) probability as the protocol executes, thus enabling relative finality with time.

4.3.1.5 Relative Finality

To establish *relative finality*, we define *k-finality* (referred to as finality) as the property of a blockchain protocol. We say the protocol has finality with parameter k, if all the honest nodes can confirm a block B, once k valid blocks have extended the chain after block B. Similar to Bitcoin and Ouroboros, we establish relative finality, i.e., the confirmation of a block by an honest node implies that with probability $1 - \eta$ ($0 < \eta < 1$), no other honest node will ever in the future disagree with the confirmed block's placement in the ledger.

To violate k-finality, the adversary must show a chain, C', that makes the honest nodes replace their CSR selected chain, C, where C and C' differ by at least k blocks.

4.3.1.6 Performance metrics

We define *transactions per second* (tps) of a protocol, as the maximum number of transactions that the protocol can add to its record every second. We define *time to finality* (t_f) as the time it takes for the protocol to confirm a block once published, given a certain required assurance level of $1-\eta$, $0 < \eta < 1$. Please note that $t_f = k \cdot t_{sl}$, where k is the *common prefix parameter* and t_{sl} is the slot length in units of time.

4.3.1.7 Requisites for a Consensus of a Blockchain Protocol

A blockchain is, in essence, a transaction ledger. For a protocol to implement a robust transaction ledger, it must satisfy two properties [28]:

- *liveness*: Once a node broadcasts a transaction, it will eventually be included in the transaction ledger and be confirmed.

- *persistence*: Once an honest node confirms a transaction, then all the other honest nodes, when queried, will agree with its placement in the ledger.

The authors of [40, 56] showed that liveness and persistence are equivalent to *common prefix*, *chain growth* and, *chain quality* for any blockchain protocol. For a PoS-based blockchain protocol that utilizes slots, these three properties are defined as follows in [41].

- *Common prefix*: We say that the protocol satisfies common prefix property with parameter k, if given the adopted chains C_1 and C_2 of any two honest nodes n_1 and n_2 at slots l_1 and l_2 respectively such that $l_1 < l_2$, then by removing k blocks from the end of C_1 we should get the prefix of C_2.

- *Chain growth*: We say that the protocol satisfies chain growth with parameter ζ, if given the adopted chains C_1 and C_2 of any an honest node n at slots l_1 and l_2 respectively such that $l_1 < l_2$, then $len(C_2) - len(C_1) \geq \zeta \cdot (l_2 - l_1)$.

- *Chain quality*: We say that the protocol satisfies chain quality with parameter υ if given a consecutive run of l blocks on a chain C adopted by an honest node, has at least $\upsilon \cdot l$ blocks generated by honest nodes.

4.3.1.8 Histogram matching

Histogram matching is a technique by which, the domain of a function is re-mapped such that its cumulative distribution function (*cdf*), F_i, matches a target *cdf*, F_t. To do this, for each value, v_t, in the domain of F_t, we find a value, v_i, in the domain of F_i, such that $F_i(v_i) = F_t(v_t)$. Using this we find the function, $F_{hm}(v_i) = v_t$, and apply it to all v_i. For details please refer to [65].

Now in the next subsection, we describe the framework that the Ouroboros family of PoS protocols deploy.

4.3.2 Framework of The Ouroboros Protocol

We briefly describe the functionalities used in the Ouroboros protocols. We use these functionalities along with our novel block power driven BPSM and CSR to achieve drastically better results. To focus on the novelty and avoid redundancy, we do not discuss the details, nuances, and implementations of these functionalities, which are well presented in the Ouroboros protocol papers.

The Ouroboros framework executes the protocol in a sequence of time periods called *epochs* that are further divided into *slots*, using the *Network Time Protocol* (NTP) (Also used by Algorand). Each epoch contains a predefined number of slots. Each slot is of a predefined length, t_{sl} ($t_{sl} > 0$). The slot number, l, (counted from the start of the blockchain execution, with the slot corresponding to the genesis block being slot $l = 0$), uniquely identifies a slot, as well as the epoch ep that the slot l is a part of. At the start of each epoch, ep, there is a *pseudo-genesis block*, $B_{PG}^{ep} = \{seed^{ep}, \{\{pk_0, r_0^{ep}\}, \{pk_1, r_1^{ep}\}, \ldots\}\}$.

It enlists the stake distribution, $\{\{pk_0, r_0^{ep}\}, \{pk_1, r_1^{ep}\}, \ldots\}$, of all the stakeholders, as well as the seed randomness, $seed^{ep}$, that is used to determine the SBP in this epoch, ep. The stake distribution of the stakeholders is as per the last block of the second last epoch, i.e., $ep - 2$. The seed randomness, is a result of a multiparty computation, *private verifiable secret sharing* (PVSS) [59] (although in practice Cardano uses *SCRAPE: SCalable Randomness Attested by Public Entities* [13]). This seed randomness is based on the input of the SBP of the previous epoch, i.e., $ep - 1$.

As shown in Fig. 4.1, this framework of epochs, slots, and pseudo-genesis blocks helps in restricting the power of the adversary. The adversary should not know the resulting random seed while it can influence it. If this is not the case, the adversary can always simulate the most favorable way to influence the random seed. Also, the adversary should not be able to change its stake distribution amongst its nodes after observing the random seed. If this is not the case, then the adversary can distribute its stake optimally based on the random seed to yield maximum effectiveness out of the BPSM.

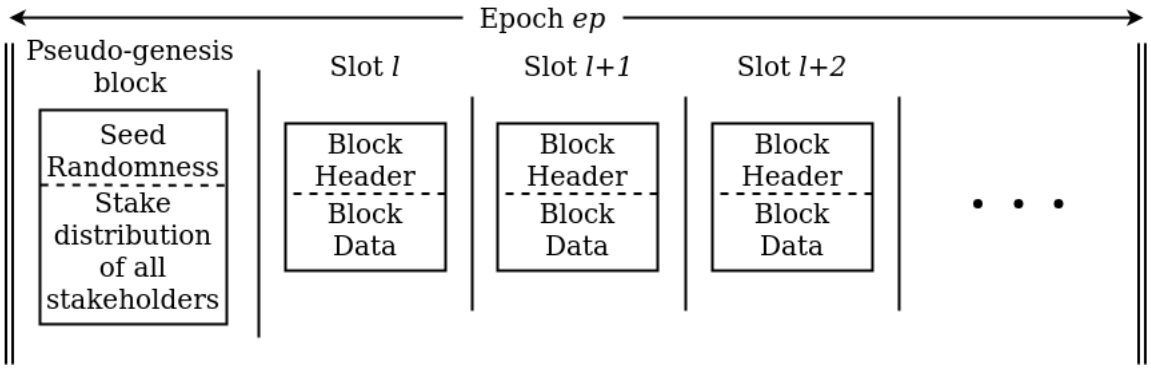


Figure 4.1: Epochs and Slots in Ouroboros

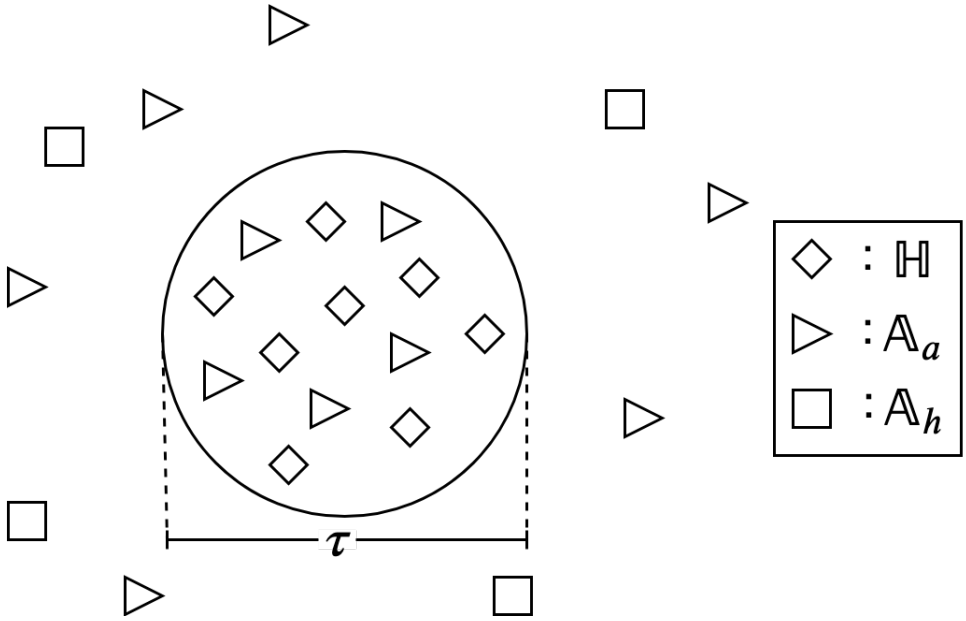


Figure 4.2: The Network N

To stochastically select SBPs, Ouroboros uses non-ideal VRFs, such as digital signatures to obtain *Verifiable Random Function* (VRF). The VRF, $\{\sigma_{uro}^{i,l,seed^{ep}}, \sigma_{proof}^{i,l,seed^{ep}}\} \leftarrow VRF(sk_i^l, l, seed^{ep})$, takes as input, the secret key of the node i corresponding to slot l, the current slot number l, and the seed randomness of epoch, ep; it produces, $\sigma_{uro}^{i,l,seed^{ep}}$, the uniform random output, of length κ in bits, and $\sigma_{proof}^{i,l,seed^{ep}}$, the proof. Any entity can verify the legitimacy of $\sigma_{uro}^{i,l,seed^{ep}}$, using $VRF_{VERIFY}(\sigma_{uro}^{i,l,seed^{ep}}, \sigma_{proof}^{i,l,seed^{ep}}, pk_i, l, seed^{ep})$. Please note, the result of a digital signature can be treated as a non-ideal VRF. However, to avoid manipulation of the probability of selection through BPSM, an *ideal-VRF* is required. This is achieved using the 2-Hash-DH [37] construction, as presented in Praos.

We use *forward secure key signatures* [4, 34, 20], as used in Praos. Forward secure key signatures, in essence, use a *public key*, pk_i, and *master secret key*, msk_i, pair for each node i. This pk_i does not change. The msk_i is used to generate a temporary secret key sk_i^l (starting from $l = 0$), which can verifiably be used only for slot l by node i, after which sk_i^{l+1} is generated and sk_i^l is *irrecoverably* destroyed. Forward secure key signatures prevent the adversary from corrupting an honest node to republish a block in its favor after the honest node has published a block in a particular slot. In this attack, the adversary will know exactly which nodes to corrupt, as they would have already published a block with a certain block power in a certain slot, thus making it public knowledge that they can achieve that block power in that slot.

Finally, in *QuickSync*, as in v1 and Praos, *moving checkpoints* are used to prevent long-range attacks such as *stake-bleeding attacks* [29]. *Moving checkpoints* are states of the blockchain, appropriately far enough into the history, such that no honest nodes would disagree with them. These moving checkpoints are assumed to be reliably and convincingly communicated to all the honest nodes joining the execution of the protocol (or any other node that has not witnessed the evolution of the honestly maintained public blockchain to establish the checkpoint for itself).

We summarize these concepts and notation in Table 6.2. Before explaining our approach that builds upon the Ouroboros framework, we first describe our network model, the adversary, and the assumptions.

4.4 Model

4.4.1 Communication Network

- We consider a network, N, of honest nodes $\mathbb{H}$, having $r_h \geq 1/2$ of the total relative stake.[3]

- The maximum communication delay (block propagation delay) between any two nodes is at most τ. In *QuickSync*, t_{sl} is set equal to τ. This implies *synchronous communication* amongst the nodes in N, w.r.t. time slots. Note, that this is similar to the communication requirement in v1, Praos and Algorand (Refer to Section 4.9 for further details).

[3]For the ease of reference we refer to nodes that are a part of N, as nodes that are in N.

- All nodes not in N, are considered to be adversarial who represent r_a relative stake.
- Thus, the adversary, $\mathbb{A}$, can be said to be comprised of nodes with malicious intent, $\mathbb{A}_a$, and nodes that follow the protocol but are not in N due to their high communication delay, $\mathbb{A}_h$. The network is described in Fig. 4.2.

4.4.2 The Adversary

We use a dynamic (*mobile* [66]) byzantine adversarial model (which in terms of ability, due to its dynamic nature, is more powerful than the one considered in v1, and the same as the ones in Praos and Algorand). The dynamic nature of the adversary is what enables *fully adaptive corruptions* (FACs). FAC is the ability of the adversary to corrupt any node instantaneously. FACs threaten the security of the protocol when the adversary can know that corrupting a certain portion of the relative stake is better than corrupting another portion of the relative stake (of the same size). The adversary, however, is always bound by $r_h \geq 1/2$.

In favor of the adversary, we assume that $\mathbb{A}_h$ is also a part of the adversary, as well as $\mathbb{A}_a$. Thus, we define the adversary, $\mathbb{A}$, as $\mathbb{A} = \mathbb{S} \setminus \mathbb{H}$. In our model, any $a \in \mathbb{A}$ can:

- read all communication between all nodes instantly.
- show any chain, that it is aware of, to any honest node.
- corrupt any node (turn any honest node into an adversarial node) at any given moment provided $r_h \geq 1/2$.
- freely, privately, and instantly communicate amongst all its nodes. All the nodes in $\mathbb{A}$ are assumed to be united by a single objective, in favor of the adversary. Hence, the adversary is considered to be a single entity.

4.4.3 Assumptions

We make the following assumptions:

- The authors of Ouroboros do not explicitly state the block size or the number of transactions per block. So, for a fair comparison, we assume that a block of the Ouroboros protocol is similar to that of the Bitcoin protocol and that of *QuickSync* in terms of size. That is to say, a block of the Ouroboros protocol, the Bitcoin protocol, and *QuickSync*, having the same tpb (typically we assume $tpb = 2000$), take about the same time to propagate.
- Given the current state of the internet, it is safe to say that a typical block, similar to Bitcoin's, consisting of 2000 transactions, reaches 95% percent of the nodes in 40 sec. [23, 7] ([21] does not discuss the average block size or the number of transactions per block). For the ease of analysis,

we ignore the 5% tail and say that within 40 sec all nodes hear of the block, i.e., that 40 sec is our upper bound on propagation, i.e., $\tau = 40$ sec.[4]

- The block *building* time is assumed to be negligible compared to τ.
- As is typical in the analysis of blockchain protocols, we assume that *honest behavior* of the honest nodes is motivated and ensured by the reward schemes. Here, by honest behavior, we mean that under *any* circumstance, the honest nodes follow the prescribed protocol without any deviation.
- In our analysis of tps, for the ease of comparison, we assume that the adversary and all active honest nodes publish blocks with their block data, at every opportunity they get. In light of this assumption, we can claim that, $tps = \frac{tpb \times \zeta}{t_{sl}}$.

4.5 *QuickSync*

First, we introduce our protocol, *QuickSync*. We then describe *QuickSync* in detail, starting with the block publisher selection mechanism (BPSM), which in turn depends upon the chain selection rule (CSR) for *QuickSync* (Section 4.5.2). Then we explain how the block publisher builds and broadcasts the block (Section 4.5.4). The last step in the protocol is block confirmation (Section 4.5.5).

4.5.1 *QuickSync*

QuickSync builds on the Ouroboros framework to attain greatly improved performance, while being resistant to a dynamic byzantine adversary with up to 50% stake. To achieve this, *QuickSync*, utilizes the novel concept of *block power* (P). Block power is a numerical metric that is defined for every block, can be computed from the block header alone, and does not depend on the block data. The block power is calculated using the relative stake, r and VRF output, σ_{uro}. The block power utilizes a special function introduced in this work, to be Sybil resistant. To determine the chain that must be extended, the nodes use the CSR. The CSR in *QuickSync* is simply to pick the chain that has the maximum *chain power*. The chain power of a chain is calculated by the summation of the block powers of all blocks of that chain.

At the start of each slot, every node publishes a block, extending the chain selected by the CSR. Since all nodes publish blocks and the block that is to be extended upon is determined by the CSR, the SBP for a slot can be said to be determined by the CSR in the next slot. Hence, the BPSM depends on the CSR. To ensure that the execution of the protocol w.r.t. slots is respected, only chains of a certain (for a given slot) length are considered valid. The valid length of a chain to be extended in the slot, l, is $l - 1$, i.e., a chain C, must be of length, $len(C) = l - 1$ to be considered by the CSR.

[4] We believe τ can be much less than 40 sec as the advent of technology. The lower the τ, better the security (w.r.t. a given t_{sl}).

For the optimal performance of *QuickSync*, we scale the relative stake, r, by a constant (parameter defined in the genesis block), s ($s > 0$), called the scale factor, to yield stake power α, which is then used to determine the block power.

Please note: We overload the notation, $P(\cdot)$, to denote block power as well chain power; When a block (chain), B_1, has higher block (chain) power than another block (chain), B_2, we say that block (chain), B_1, is better than block (chain), B_2; We use the notation B^l_C, to refer to the block, B, that is a part of chain, C, that was published in slot l.

4.5.2 Block Publisher Selection Mechanism and Chain Selection Rule

We propose the following CSR; to be followed by node, i, in slot, l, to pick a chain, C_{csr}, from a set of chains known to node i at the start of slot l, $S^{Chains}_{view(i,l)}$, to be extended by publishing a block in slot l:

1. From the set of chains, $S^{Chains}_{view(i,l)}$, select a subset of chains $S^{validChains}_{view(i,l)}$, such that $\forall C \in S^{validChains}_{view(i,l)}, len(C) = l - 1$.
2. Calculate the chain power $P(C)$, of every chain C in $S^{validChains}_{view(i,l)}$. The chain with the maximum chain power is the C_{csr}.
3. Publish a block extending chain C_{csr}.

Since 2^κ is typically a very large number, the probability that two chains have the same chain power is clearly very low. In the event that this does happen, the next SBP will extend one of them, all nodes in N will then accept this extension (This is very similar to the attack wherein the adversary provides two different block data with similar block headers. Please refer to Section 4.8).

4.5.3 Calculating Block Power and Chain Power

To calculate the chain power, $P(C)$, of a chain, C, sum the block powers, $P(B)$, of all the blocks $B \in C$.

To calculate the block power, $P(B^l_i)$, of a block B^l_i:

1. Calculate the stake power, α^{ep}_i, of node i in epoch ep, by multiplying r^{ep}_i by the scale factor, s.
2. Normalize $\sigma^{i,l,seed^{ep}}_{uro}$ by dividing it by 2^κ, obtaining $\sigma^{i,l,seed^{ep}}_{nuro}$ which has a range of $[0, 1]$.
3. Map the value $\sigma^{i,l,seed^{ep}}_{nuro}$, from the domain of a uniform *pdf* with range $[0, 1]$, to the corresponding domain of the function $f_{\alpha^{ep}_i}(x) = \alpha^{ep}_i x^{\alpha^{ep}_i - 1}$, using histogram matching.

4.5.4 Block Publishing

Block publishing comprises of building a block and then broadcasting it. To publish a block in slot, l, a node, i, must build the block data, Bd^l_i, and the block header, Bh^l_i, as prescribed below. Once the

block, $B_i^l = \{Bh_i^l, Bd_i^l\}$, is built, it is broadcast, completing the process of publishing block, B_i^l. In *QuickSync*, to optimize the communication process, only the best block seen is propagated forward by the nodes instead of their own block.

Block building Block building is done at the start of the block, after the chain C_{csr} has been selected by the CSR as the one to be extended. At the start of slot l, the node i, collects transactions, $\{tx_0, tx_1, \ldots\}$, received by the end of slot $l-1$, that have not yet been added to the blockchain and forms the block data, $Bd_i^l = \{tx_0, tx_1, \ldots\}$. The block header is then built to the format, $Bh_i^l = \{pk_i, r_i^{ep}, l, \{hash(B_{C_{csr}}^{l-1}), null(B_{C_{csr}}^{l-1})\}, \{\sigma_{uro}^{sk_i^l, l, seed^{ep}}, \sigma_{proof}^{sk_i^l, l, seed^{ep}}\}, MTR(Bd_i^l)\}$; where i is the node with public key, pk_i, and relative stake, r_i^{ep}; l is the slot number; ep is the epoch number; $\{\sigma_{uro}^{sk_i^l, l, seed^{ep}}, \sigma_{proof}^{sk_i^l, l, seed^{ep}}\}$ is generated by $VRF(sk_i^l, l, seed^{ep})$; $MTR(Bd_i^l)$ is the *Merkle tree root* of the block data Bd_i^l, it is what binds the block header to the block data Bd_i^l (also, fixes order of tx in Bd_i^l); $\{hash(B_{C_{csr}}^{l-1}), null(B_{C_{csr}}^{l-1})\}$ binds block B_i^l as extending the chain C_{csr} beyond block $B_{C_{csr}}^{l-1}$, and denotes whether the block $B_{C_{csr}}^{l-1}$, was a null block (Please refer to Section 4.8).

The block, B_i^l, is now ready to be broadcast. Note, the block header is sent along with a digital signature over itself for sender authentication.

Broadcasting the best block Since all nodes will have valid blocks that are contenders for the best block and can be selected to be built on in the next slot, waiting to collect all blocks from all users in N will require τ to be very high. So instead, the nodes only download the block data of the best block header seen yet. Thus, any node will only download and broadcast the block data of the best block header seen so far. From the perspective of block propagation time, this method of communication is equivalent to communicating a single valid block. Since the block header is sufficient to determine the block power of a given block, a node is aware of a block and its power as soon as it receives its block header. Once a node is aware of a block, it may or may not attempt to download its block data. An honest node should always try to propagate the best block header it is aware of, to all other nodes, at any given moment.

Consider a network, as shown in Fig. 4.3. W.L.O.G., let us say that the node n_0 is the one that wants to build the next block and hence needs to know which is the best block in the current slot. The nodes $n_0, n_1, n_2, n_3, n_4, n_5$ have published blocks, $B_j = \{Bh_j, Bd_j\}; j = \{0..5\}$ with $P(B_1) < P(B_2) < P(B_3) < P(B_4) < P(B_5)$. Say, node n_0, sees the blocks in the following order: B_1, B_4, B_3, B_5, B_2. Now, first, n_0 sees Bh_1, and since it is better than Bh_0, and hence the best block it is aware of at the moment, begins to download Bd_1. Whether or not n_0 has finished downloading Bd_1, once it sees Bh_4, it will stop downloading Bd_1 and start downloading Bd_4. n_0, will, however, disregard B_3, as it already is aware of a better block, i.e., B_4. However, when n_0 hears of Bh_5, it will switch to downloading Bd_5 instead, as its block power is higher. Again, it will disregard B_2, as it is aware of a better block B_5.

While broadcasting blocks has been discussed here, the nodes in fact broadcast chains, in essentially the same manner as discussed above. This broadcasting of chains devolves to broadcasting blocks

when the chains differ only by one block, i.e., the one published in slot l. We omit the discussion on broadcasting chains, i.e., the ones that differ from C_{csr}, by more than one block for simplicity and tractability. However, it is important to note that the nodes always propagate the best chain seen so far (similar to the best block propagation as discussed above).

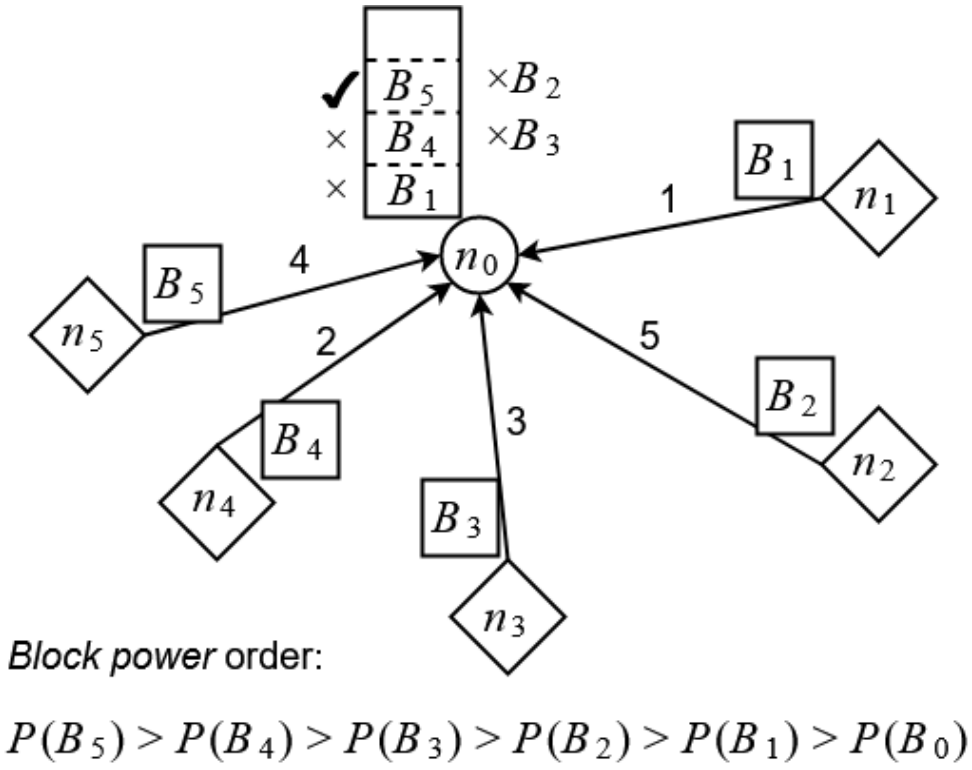


Figure 4.3: Block download stack of a node.

4.5.5 Block Confirmation

The nodes confirm all, but the most recent k blocks, of the chain selected by the C_{csr}. This confirmation is consistent with *k-finality* that is attained by *QuickSync*.

We conclude the presentation of *QuickSync* with its pseudo-code, given in Fig. 4.4, and the power computation pseudo-code, given in Fig. 4.5. The intuition behind *QuickSync* is discussed in Section 4.7. The power of our approach can be summarized as follows: The nodes collectively, as common knowledge, know which chain to adopt, which then becomes the honest chain. As soon as a fork is witnessed, adversarial or not, the nodes know whether or not to adopt it, the nodes are not confused, as they would be in protocols that do not differentiate between published blocks, such as *Bitcoin* or Ouroboros. However, the adversary can still attempt to fork and develop chains and use them to violate finality. In the next section, we prove that *QuickSync* is secure against such attempts.

Figure 4.4: *QuickSync* Protocol

***QuickSync Protocol Pseudo-code*;**

Followed by node i in slot l:
INPUT: $\{sk_i^l, r_i^{ep}, seed^{ep}, S_{view(i,l)}^{Chains}, s, \{tx_0, tx_1, \ldots\}\}$
Step 1: *Chain selection*

1: From $S_{view(i,l)}^{Chains}$, select the subset
$S_{view(i,l)}^{validChains} : \forall C \in S_{view(i,l)}^{validChains} | len(C) = l - 1.$
2: $\forall C \in S_{view(i,l)}^{validChains}$, calculate, $P(C)$.
3: Select the chain,
$C_{csr} : C_{csr} = \text{argmax}_{C \in S_{view(i,l)}^{validChains}} P(C)$

Step 2: *Block publishing*

a) *Block building*

1: Build $Bd_i^l = \{tx_0, tx_1, \ldots\}$, and obtain $MTR(Bd_i^l)$.
2: $\{\sigma_{uro}^{i,l,seed^{ep}}, \sigma_{proof}^{i,l,seed^{ep}}\} \leftarrow VRF(sk_i^l, l, seed^{ep})$
3: Obtain $\{hash(B_{C_{csr}}^{l-1}), null(B_{C_{csr}}^{l-1})\}$
4: Build
$Bh_i^l = \{pk_i, r_i^{ep}, l, \{hash(B_{C_{csr}}^{l-1}), null(B_{C_{csr}}^{l-1})\}, \{\sigma_{uro}^{sk_i^l,l,seed^{ep}}, \sigma_{proof}^{sk_i^l,l,seed^{ep}}\}, MTR(Bd_i^l)\}$

b) *Block broadcasting*

1: Set $C_{broadcast} = \{C_{csr}, B_i^l\}$
2: **while** Current Slot $== l$ **do**
3: Listen and receive C_{rec} from other nodes
4: **if** $P(C_{rec}) > P(C_{broadcast})$ **then**
5: Set $C_{broadcast} = C_{rec}$
6: **end if**
7: Broadcast $C_{broadcast}$
8: **end while**

Step 3: *Block confirmation*

1: **for** $j > 0; j++; j \leq len(C_{csr}) - k$ **do**
2: Confirm block, B_C^j.
3: **end for**

Figure 4.5: Block and Chain Power Computation

Chain Power $P(C)$

INPUT: Chain C
OUTPUT: $P(C)$

1: $Sum = 0$
2: **for** $l > 0; l++; l \leq len(C)$ **do**
3: $\quad Sum+ = P(B_C^l)$
4: **end for**
5: $P(C) = Sum$

Block Power $P(B_C^l)$

INPUT: Block B_C^l
OUTPUT: $P(B_C^l)$

1: From the header of block B_C^l, obtain $pk_i, r_i^{ep}, \sigma_{uro} = \sigma_{uro}^{sk_i^l, l, seed^{ep}}$, where i is the publisher of block B_C^l.
2: From the genesis block obtain s.
3: $\alpha_i^{ep} = r_i^{ep} \times s$
4: $\sigma_{nuro} = \frac{\sigma_{uro}}{2^\kappa}$
5: $P(B_C^l) = \sqrt[\alpha_i^{ep}]{\sigma_{nuro}}$

4.6 Security Analysis

To prove the security of *QuickSync*, w.r.t. *liveness* and *persistence*, we need to establish the three prerequisite properties of a blockchain protocol, i.e., common prefix, chain growth, and chain quality, as given in Section 4.3.1.7. We do this analysis in Section 4.6.1 and further discuss why *QuickSync* is resilient to FACs.

QuickSync does not require a *Forkable Strings* analysis, as presented in v1 and Praos. Due to the elegance and naturality of our approach, *QuickSync* avoids the inherent complexities of the Ouroboros protocols. All observed (public) forks are immediately and trivially resolved using chain power as the nodes will only build on the chain with the highest chain power. If an adversarial chain with higher chain power than an honest chain is revealed before it can violate finality, that adversarial chain becomes the new honest chain. To violate finality (and hence common prefix), an adversarial chain must have higher chain power than the honest chain while forked for at least k blocks; the probability of this is exponentially bounded with k, as proved in this section.

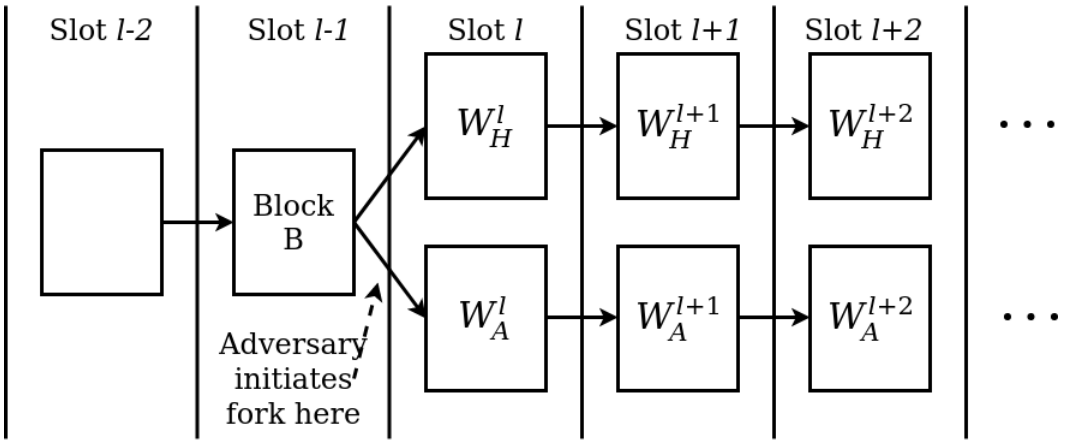


Figure 4.6: The forking attempt by the adversary to violate finality

4.6.1 Analysis of Security Requisites

Claim. Let $X_1, X_2, \ldots, X_M$ be independent random variables with zero mean, such that $|X_l| \leq K \forall l \in [M], K > 0$. Also, $E[{X_l}^2] \leq \sigma_l^2$. Then, $\forall \lambda > 0$:

$$\Sigma_{M=k}^{\infty} Pr\left(\frac{1}{M}\Sigma_{l=1}^{M} X_l \geq \lambda\right) \leq \frac{e^{-c\cdot k}}{1-e^{-c}} \text{ where, } c = \frac{\lambda^2}{2(\sigma_l^2 + K\lambda/3)}$$

Proof. Using *Bernstein's inequality* [17] for $X_1, X_2, \ldots, X_M$,: $\forall \lambda > 0$,

$$Pr\left(\frac{1}{M}\Sigma_{l=1}^{M} X_l \geq \lambda\right) \leq \exp(\frac{-M\lambda^2/2}{\sigma_l^2 + K\lambda/3})$$

$$\implies \Sigma_{M=k}^{\infty} Pr\left(\frac{1}{M}\Sigma_{l=1}^{M} X_l \geq \lambda\right) \leq \frac{e^{-c\cdot k}}{1-e^{-c}} \text{ where, } c = \frac{\lambda^2}{2(\sigma_l^2 + K\lambda/3)}$$

□

Lemma. The probability that *QuickSync* does not satisfy the common prefix property with parameter k is given by:

$$\varepsilon_{cp} \leq \frac{Le^{-ck}}{1-e^{-c}} \text{ where, } c = \frac{\lambda^2}{2(\sigma_l^2 + K\lambda/3)}$$

Proof. Let E_1 be the event when the adversary holds a chain that forks by more than k blocks and has higher chain power w.r.t. the honest chain, in the lifetime of the protocol. When this event occurs, the adversary can show its chain to the honest nodes, making them accept it. When the honest nodes accept a chain that forks from their own by more than k blocks, they are forced to change a confirmed block. This violates finality, and in turn, violates common prefix. To violate finality, w.r.t. a particular block B', the adversary must fork the chain at any point before the block B', say at block B and then have a better chain after at least k blocks from B. Let E_1^B denote the event that finality is violated with a fork starting at a certain block B.

Now we establish the probability, $\eta = Pr(E_1^B)$, that the adversary can violate finality with a fork starting at block B. Since this fork can start at any point in the lifetime of the protocol, we can say that the probability, $\varepsilon_{cp} = Pr(E_1)$, that common prefix is ever violated is at most $L \times \eta$, where L is the lifetime of the protocol in slots.

To calculate the upper bound on η, we use Bernstein's inequality. Using this, we bound the mean of the power of the blocks from a given block, say B. If at any point, after k blocks from block B, the mean power of the adversarial chain exceeds that of the honest chain then, we say event E_1^B occurs violating finality.

To use Claim 4.6.1, we take:

- $W_A^l = \alpha_A x^{\alpha_A - 1}$ and $W_H^l = \alpha_H x^{\alpha_H - 1}$ to be the random variables, representing the power of the blocks of the adversary and the honest nodes respectively, in slot l. Here, $\alpha_A = s \times r_a$ and $\alpha_H = s \times r_h$. Please refer to Fig. 4.6.
- $X_l = W_A - W_H + (E(W_H) - E(W_A))$.
- $\lambda = E(W_H) - E(W_A) = \frac{\alpha_H}{\alpha_H+1} - \frac{\alpha_A}{\alpha_A+1} > 0$, representing the advantage of honest nodes.
- $K = 1 + \left(\frac{\alpha_H}{\alpha_H+1} - \frac{\alpha_A}{\alpha_A+1}\right)$
- $\sigma_l^2 = \left(\frac{\alpha_A}{\alpha_A+2} - \left(\frac{\alpha_A}{\alpha_A+1}\right)^2\right) + \left(\frac{\alpha_H}{\alpha_H+2} - \left(\frac{\alpha_H}{\alpha_H+1}\right)^2\right)$

Thus:

- $E(X_l) = 0$
- $|X_l| \leq 1 + (E(W_H) - E(W_A))$
  $= 1 + (\alpha_H/(\alpha_H + 1) - \alpha_A/(\alpha_A + 1)) = K$

- $E[X_l]^2 = \sigma^2(W_A - W_H + (E(W_H) - E(W_A)))$
  $= \sigma^2(W_A) + \sigma^2(W_H)$, since $E(X_l) = 0$
  $= \left(\frac{\alpha_A}{\alpha_A + 2} - \left(\frac{\alpha_A}{\alpha_A + 1}\right)^2\right) + \left(\frac{\alpha_H}{\alpha_H + 2} - \left(\frac{\alpha_H}{\alpha_H + 1}\right)^2\right)$

Therefore by Claim 4.6.1, $\eta \leq \frac{e^{-ck}}{1-e^{-c}}$, where, $c = \frac{\lambda^2}{2(\sigma_l^2 + K\lambda/3)}$

Hence, the probability that common prefix is violated is, $\varepsilon_{cp} \leq \frac{Le^{-ck}}{1-e^{-c}}$, where, $c = \frac{\lambda^2}{2(\sigma_l^2 + K\lambda/3)}$

Thus common prefix property is established. □

As shown in Fig. 4.7a the plot of practical t_f vs s resembles an elbow curve. That is because as s increases, even though the difference between the expectation, of the block power of the honest nodes and the block power of the adversary, decreases, the variance too, decreases. This effect tapers out after $s = 8$. Thus, we choose $s = 8$ for the best practical finality.

Refer to Fig. 4.7b for the error probability (violation of finality) η as a function of k.

Proposition 3. *In* QuickSync*, a block is added to the chain in each slot, provided* $r^{active} > 0$.

Proof. In *QuickSync*, the BPSM depends on the CSR. The CSR used in *QuickSync* assigns a chain power for each valid chain and select the one with the highest power, and hence will be able to select a chain as long as there is atleast one valid chain. To have a valid chain, for the consideration by the CSR, in the current slot, even one block is sufficient to have been published in the previous slot. Hence, even if there is just one active node in a given slot, there will be a block added to the chain in that slot.

Lemma. *QuickSync* satisfies the chain growth property with parameter $\zeta = 1$, given that $r^{active} > 0$.

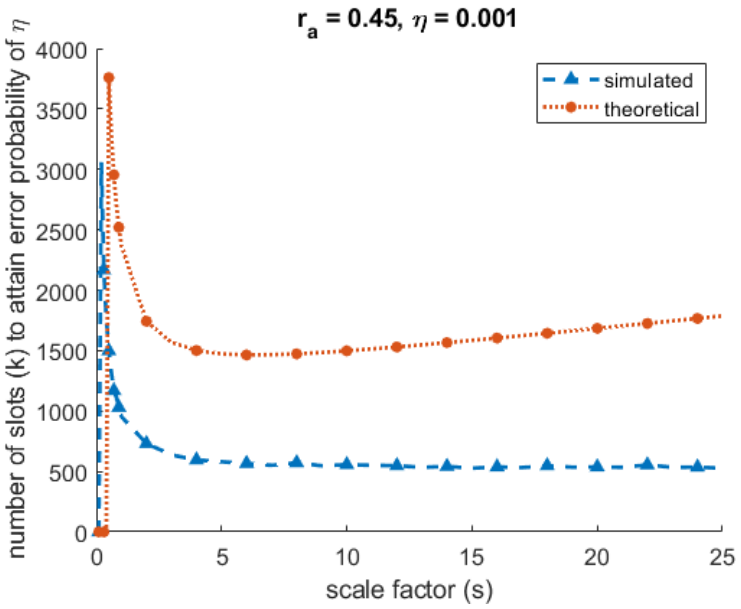
Proof. Let E_2 be the event where there is no block added to the chain in a slot, in the lifetime of the protocol. To satisfy chain growth with parameter $\zeta = 1$, a block must be added to the chain in each slot, i.e., $Pr(E_2) = 0$. For this, the BPSM must select a node in each slot.

However, by Proposition 3, $Pr(E_2) = 0$ if $r^{active} > 0$ for *QuickSync*. Therefore *QuickSync* satisfies the chain growth property with parameter $\zeta = 1$, provided $r^{active} > 0$. Thus, the chain growth property is established. □

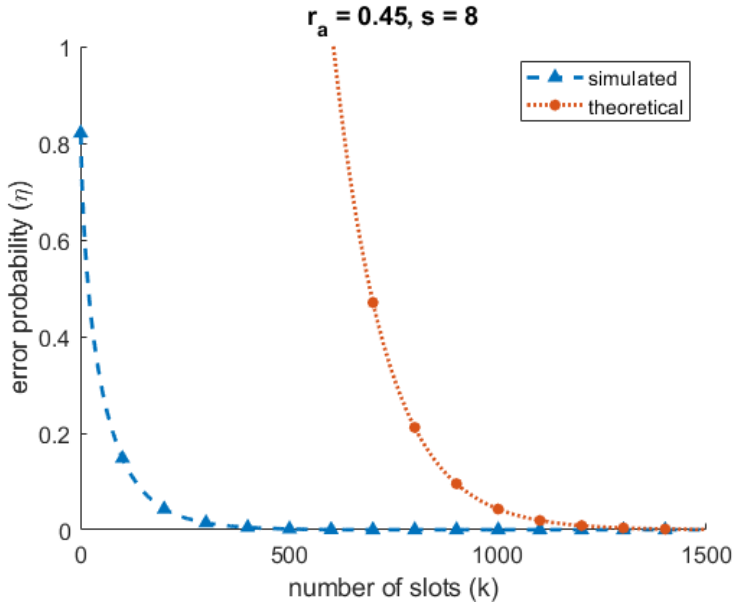
Lemma. The probability that *QuickSync* does not satisfy the chain quality property with parameter $\upsilon = 1/k$ is ε_{cp}. *Proof.* Let E_3 be the event that there is a run of k or more consecutive blocks, published by the adversary, on the chain, C, held by an honest node, in the lifetime of the protocol. Let E_3', be the event where the sum of power of k consecutive adversarial blocks is higher than the sum of power of the best corresponding honest blocks, in the lifetime of the protocol. Clearly, E_1 subsumes E_3' (since if E_3' occurs; E_1 too must occur). Now, for E_3 to occur, E_3' must occur. Thus, $Pr(E_1) \geq Pr(E_3)$.

Now, evidently, when E_3 does not occur, there is at least one honest block in every run of k consecutive blocks. Hence, when E_3 does not occur, $\upsilon >= 1/k$. Therefore, the probability that chain quality with parameter $\upsilon = 1/k$ is violated is at most ε_{cp}.

Thus, the chain quality property is established. □



(a) Finality as a function of s



(b) Error probability η as a function of k

Figure 4.7: Simulation and Theoretical Analysis

Theorem 1. *The probability that any of; common prefix with parameter k, chain growth with parameter $\zeta = 1$, chain quality with parameter $\upsilon = 1/k$ are violated in the lifetime of the protocol, thereby violating liveness and persistence is:*

$$\varepsilon_{lp} \leq 2\varepsilon_{cp}$$

Proof. The above theorem follows from Lemmas 1, 2, 3 and the union bound $Pr(E_1 \cup E_3)$. □

Proposition 4. QuickSync *is resilient to* fully adaptive corruptions *(FACs).*

Proof. For a protocol to be resilient to FACs, it must be resilient against attempts of the following two corruptions:

- *Posterior corruption*: It is the adversary's ability to corrupt a node, use its stake and hence its block power, to publish a block in a past slot (w.r.t. the current slot).
- *Anterior corruption*: It is the adversary's ability to corrupt a node that *might* have the best block in the future (w.r.t. the current slot).

QuickSync avoids posterior corruption by using forward secure key signatures, as in Praos. Although anterior corruption by itself is not an issue, if the adversary knows which nodes *will* have the best block with certainty or disproportionately high probability, then the adversary will corrupt those nodes and gain an advantage over the honest nodes. To avoid this, the adversary must not know which node will have the best block in the future. To this end, the nodes that attempt to publish a block must not reveal their block power until they publish the block. This is exactly the approach in *QuickSync*. In *QuickSync*, the block power is given in the block header, and the block header is not revealed until the block is published. Hence, the block power is not revealed until the block is published.

Thus, we say that *QuickSync* is resilient to FACs. □

Please refer to Section 4.8 for the analysis of attack strategies against *QuickSync*.

4.7 The Intuition behind *QuickSync*

We require that our protocol, *QuickSync*, satisfies the following two requirements:

- To ensure security against FACs, the SBP must not be revealed before it publishes the block.
- Forks costs performance. Hence, forking amongst the honest nodes must be avoided. Thus, at any time, ideally, we must have only one block that should be extended.

To fulfill the first requirement, the computation required by the BPSM must be done privately. For the second requirement, the BPSM must select exactly one block publisher as the SBP. Satisfying both of these requirements together is non-trivial, as we must privately, securely, provably and efficiently select

exactly one party in a multiparty weighted coin-toss with guaranteed output using a seed-randomness (which is easy to generate using PVSS or SCRAPE as mentioned earlier). The BPSM presented in this work solves for both these requirements through the CSR. In contrast, v1 solves only the second requirement but not the first, while Praos solves only the first requirement but not the second.

Each node publishes its block, in an attempt to have it selected as the next block to be added to the blockchain. However, if the nodes arbitrarily choose one of the blocks they received, they might select different blocks, hence causing forks. Suppose, if, from a given set of options, *all the nodes know which block to extend*. If this is the case, then all the nodes would extend the same block, and hence avoid forks. Now since all the nodes will publish their own blocks, and only one must be selected amongst them, in such a way that all arrive at the same result, we establish a metric called *block power*, by which all the nodes must evaluate the competing blocks. The block (chain) with the maximum block (chain) power is selected. This metric must, of course, be dependent on the stake of the block publisher so that our protocol can be PoS-based.

Naive implementations of the above approach are vulnerable to Sybil attacks. To solve for this, we present a Sybil attack resistant function:

$$f_\alpha(x) = \alpha x^{\alpha-1} \tag{4.1}$$

where x represents the block power, α is the stake power of the block publisher, and the resulting $f_\alpha(x)$ is the probability distribution function (*pdf*) of the block power of the block.

To be Sybil resistant, we must ensure that whether a node is represented as a single entity of stake α or two entities of stake power α_1 and α_2, the *pdf* of the node's effective block power (the node's maximum block power, as one or more entities) remains the same. Hence, the node has the same probability of becoming the SBP, regardless of division into smaller nodes or aggregation into bigger ones.

The function in Eqn. 4.1 is resilient to Sybil attacks due to the following property:

$$f_\alpha(x) = f_{\alpha_1}(x) \int_0^x f_{\alpha_2}(y)dy + f_{\alpha_2}(x) \int_0^x f_{\alpha_1}(y)dy$$

where $\alpha = \alpha_1 + \alpha_2 \; \forall \; \alpha_1, \alpha_2 > 0$, the attacker splitting its computing power into two entities.

To utilize $f_\alpha(x)$, we map the uniformly distributed normalized output over $[0, 1]$, σ_{nuro}, of the private computation required by the BPSM to the domain of $f_\alpha(x)$, using histogram matching, where α is the stake power of the block publisher. The re-mapped domain is then used as the *block power*.

To show why and how our block power approach is powerful, we use an example. Consider a fork, where nodes, n_a and n_b, both are aware of two blocks, B_1 and B_2 viable for extension. Now, in both Ouroboros and *Bitcoin*, there is no distinction made between the two blocks. Hence, the node, n_a, may extend block B_1, while node, n_b, extends block B_2. This causes an extension of the fork. This is what *QuickSync* gracefully avoids. The same scenario in *QuickSync* would lead to both nodes extending either block B_1 or block B_2, depending on their block powers. This collective decision to select one block that is to be extended, resolves the fork. Since the forks are immediately resolved, it is equivalent to the forks never occurring. This results in avoiding forking amongst honest nodes and is why the block power approach is so powerful.

4.8 Analysis of Attack Strategies against *QuickSync*

In this section, we discuss possible attack strategies and show that they are futile against *QuickSync*.

Sybil attack Consider two scenarios: i) in which there is a node with stake power α_0 and ii) there are two nodes with stake power α_1 and α_2. In the first scenario, there will be one value of block power in consideration, whereas, in the second scenario, there are two values. However, in the second scenario, only the higher block power is relevant as the CSR will (and hence the BPSM) select the block with the maximum block power. To avoid Sybil attack, we need that in both the scenarios, the *pdf* of the relevant block power to be the same, else the adversary has an incentive to split or aggregate stake power to have a higher probability of getting selected as SBP. To this end we must ensure that the *pdf* of the block power in the first scenario must equivalent to the *pdf* of the maximum of the two block powers in the second scenario, i.e., we need a block power function that satisfies:

$$f_{\alpha_0}(x) = f_{\alpha_1}(x) \int_0^x f_{\alpha_2}(y)dy + f_{\alpha_2}(x) \int_0^x f_{\alpha_1}(y)dy$$

where $\alpha_0 = \alpha_1 + \alpha_2 \; \forall \; \alpha_1, \alpha_2 > 0$, which is satisfied by $f_\alpha(x) = \alpha x^{\alpha-1}$.

Also the probability that a node, with stake α_1, wins over a node, with stake α_2, is:

$$\int_0^1 f_{\alpha_1}(x) \int_0^x f_{\alpha_2}(y)dydx = \frac{\alpha_1}{\alpha_1 + \alpha_2}$$

Double spending attack In this sort of an attack, the adversary attempts to replace a certain block B' on the chain of an honest node after it has confirmed the block B'. To do this, the adversary must show a better chain that forks by at least k blocks, starting from before block B', to an honest node that has confirmed block B'. This attack is ineffective, given that the common prefix property is established. However, the adversary can attempt to reduce the effectiveness of the stake power of N or increase the effectiveness of its own stake power. It can attempt to do so in the following two ways, neither of which are effective:

- *Split N attack*: Ideally, all the nodes in N choose the same chain and attempt to build on top of it. In this case, the stake power of the honest nodes is undivided. However, even if the adversary attempts to divide N, it is easy to see that the protocol will not be compromised. The adversary can try to split N by:
  - Showing different chains to different subsets of N. Say, there is a chain C, that was built by N and sent to all in N. The adversary will have to show a node, say n_1, in N, a chain better than C, say C_1. Now either the node n_1 finds the block that leads to its chain being the best chain or some other node does. If n_1 finds it extending C_1, it will broadcast the chain since it is an honest node. If any node other than n_1 finds it, the protocol executes as usual. So, even if the adversary tries to show a different chain to some node in N, N will not be worse off.

– Giving different block data with the same block header to different subsets of N. This case is a trivial subset of the above case. In fact, this can also happen in Ouroboros. If the adversary does try to give two blocks, with the same block header, but with different block data, the node which finds the next block that leads to its chain being the best chain, will extend the version it has, and all will then accept this.

- *Borrow power attack*: Consider the following case, starting from a common block B. N sees the block B_H^1 as the best block extending B. A node n in N, is shown a better block B_A^1 by the adversary. Now, say, n makes block B_n^2 extending B_A^1, rest of N makes block B_H^2 extending B_H^1, and the adversary makes block B_A^2 extending B_A^1; and if, $P(B_n^2) > P(B_A^2)$ and $P(B_A^1) + P(B_n^2) < P(B_H^1) + P(B_H^2)$; then N will use $\{B_H^1, B_H^2\}$, but now the adversary also has $\{B_A^1, B_n^2\}$ which is better than it's own $\{B_A^1, B_A^2\}$. So in a sense the adversary has *borrowed* n's power. We use simulations to show that this attack does not significantly affect the adversary's ability to violate common prefix. For details, please refer to Section 4.8.1.

Missing block data attack Since the nodes are required to build on the chain with the maximum power. The adversary (or any node not in N), when it has the best chain, could show its block header to nodes in N, and then not send the block data. This would result in the nodes in N being unable to proceed with the protocol execution and hence would stall the protocol. All honest nodes, however, will always broadcast the best block header (that they are aware of) along with its block data, and will always include the block data of the previous block when extending it.

To avoid this scenario, we allow nodes to extend blocks without their block data. We call such a block, that has only the block header and no block data, as a *null block*. This approach would keep the protocol from stalling as well as prevent the honest chain from losing power (as we can use null blocks when calculating chain power). The information that the previous block was a null block or not is given in the header of the block extending it. Since this information is immutable, given a chain, each block can be unambiguously determined to be null or not. Note that null blocks are different from blocks that have 0 transactions in their block data.

We believe the effect of this attack can be easily mitigated through reward schemes. E.g., by reducing the utility (block reward and transaction fees) of both the owner (publisher) of the null block as well as of the node extending it. As the cost of publishing null blocks is discouraging, the adversary will not launch this attack. The optimal method that can be used to deal with this scenario is dependent on the specifics of the implementation. We, however, assume that this affects neither tps nor chain growth.

4.8.1 Borrow Power Attack

Let us calculate the impact of the borrow power attack. Each time the adversarial chain is better than or equal to the honest chain, the adversary can show this chain to a fraction of the honest nodes in N, and have them build on it in that slot. When the adversary shows its chain, it may gain or lose utility,

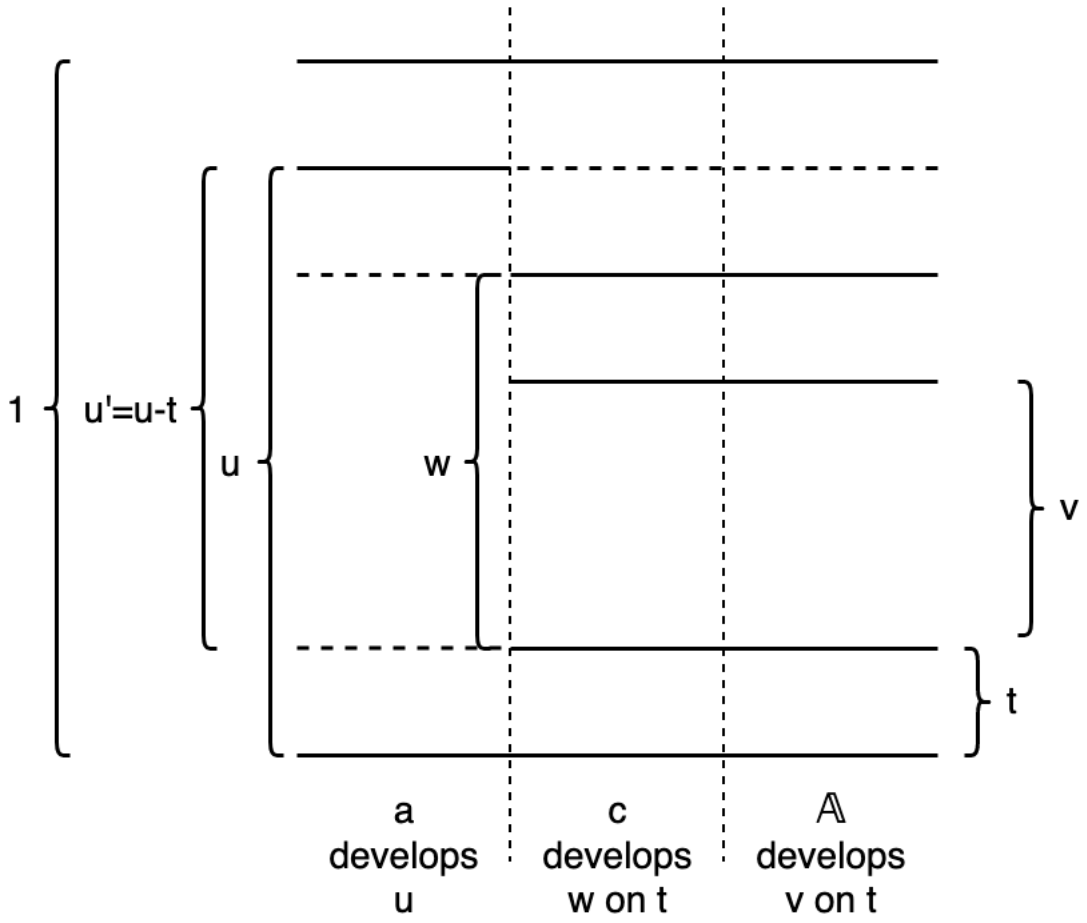


Figure 4.8: Graphical representation of power of blocks in the borrow power attack

depending on the stochastic outcome. Before the adversary shows its chain, it knows the value of power by which it beats the honest chain, as well as the power of its block in that slot. Using this information, the adversary calculates the optimum fraction of nodes to show its chain to. Here, we show that even though the expected utility is positive for the adversary, the gain is, in fact, negligible and marginally affects the adversary's ability to violate common prefix.

Let us consider the power of the chain of the honest nodes, up till the given slot, to be the baseline. Let us say that the adversarial chain has t more power than that of the honest nodes. Let v be the power of the adversarial block in this slot. We spilt stake power of $\mathbb{H}$ as $c + a$ where c is the fraction of stake power of the nodes in $\mathbb{H}$ that build on the chain that the adversary shows to them (adversarial chain) and a is the fraction of stake power of the remaining nodes in $\mathbb{H}$ which build on the chain built by the honest nodes. Here, we overload the notation c, a to also represent the corresponding sets of nodes, respectively. Let u, w denote the block powers of the blocks generated by a and c respectively, and hence u and w are random variables with *pdfs* au^{a-1} and cw^{c-1} respectively. Here, we ignore the case $v + t >= 1$, as in that case, the adversary can never expect to gain anything and hence will never play

that scenario out. Please refer to Fig. 4.8 for graphical representation. Note that all these variables have the constraints: $0 < v < 1, 0 < t < 1, 0 < u < 1, 0 < w < 1, a > 0, c > 0, v + t < 1, a + c = \alpha_h$.

Now, as mentioned above, the exact utility of the adversary depends on the power generated by a and c. The adversary selects the c that gives the optimal expected utility. The utility here refers to the gain in chain power that the adversary obtains when it conducts the attacks in the given slot. The change in the difference in chain power of N and the adversary is realized by adopting (or being forced to adopt a different chain than the one held before the attack). Whenever c develops a chain that is better than a's, N adopts it. Since the adversary needs to maintain a chain that differs from the honest chain by at least k blocks, from the one N holds, the adversary can no longer build on the chain that c has developed or the one that it showed to c (unless the adversary wants to start the fork from another point). However, the adversary can now use the chain N has discarded, and it will do so if profitable.

We use the following assumptions:
- We assume that the adversary is comprised of several small nodes. This assumption implies that the adversary has several chains close to the power of its best chain, and when N adopts its chain, it does lose any chain power. This means that whenever c makes a chain better than the one of a, and the honest nodes adopts c's chain (the one built on the adversary's best chain before the attack), thus rendering c's chain useless for the adversary (for the current forking attempt), the adversary itself does not lose any chain power.
- For simplicity, we assume that all the stake power in N is held by just one node. That is, there is only one block published by N in any slot. However, in the first slot of the forking attempt, we ignore this assumption. Instead, we assume that N is comprised of several small nodes, in favor of the adversary. The adversary can choose between any of the blocks it has built and any block the honest nodes have published except the best block of N. Due to this, the adversary, in the first slot of the forking attempt, always has a better or an equivalent block as compared to N.

We consider the following six cases. For each of these cases, we now determine the gain in chain power, i.e., the utility gained by the adversary on N when it launches borrow power attack. Note that this gain is as compared to the case where the adversary does not launch such an attack. The utility gained by N is utility lost by the adversary.

- Case 1: $v + t > w + t > u$

  If $w > u$, then N gains utility t. If $u \geq w$, then N gains utility $(w + t - u)$.

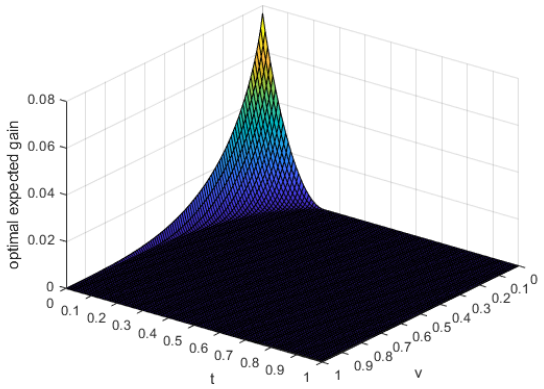
  N's expected gain: $c1_h = \int_0^v cw^{c-1} \left(\int_0^w tau^{a-1} \, \mathrm{d}u + \int_w^{w+t} (w + t - u)\, au^{a-1} \, \mathrm{d}u \right) \mathrm{d}w$

- Case 2: $v + t > u > w + t$; No one gets any utility.

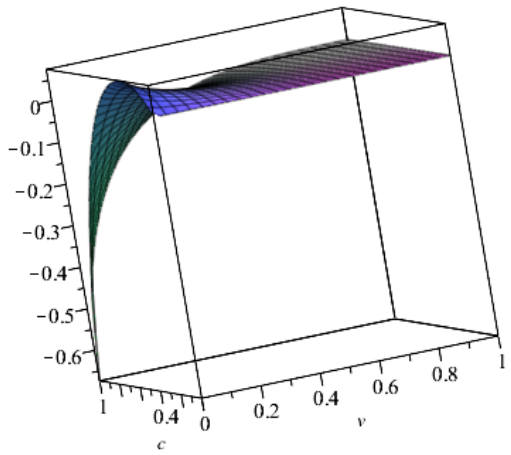
- Case 3: $u > v + t > w + t$; No one gets any utility.

- Case 4: $w + t > v + t > u$

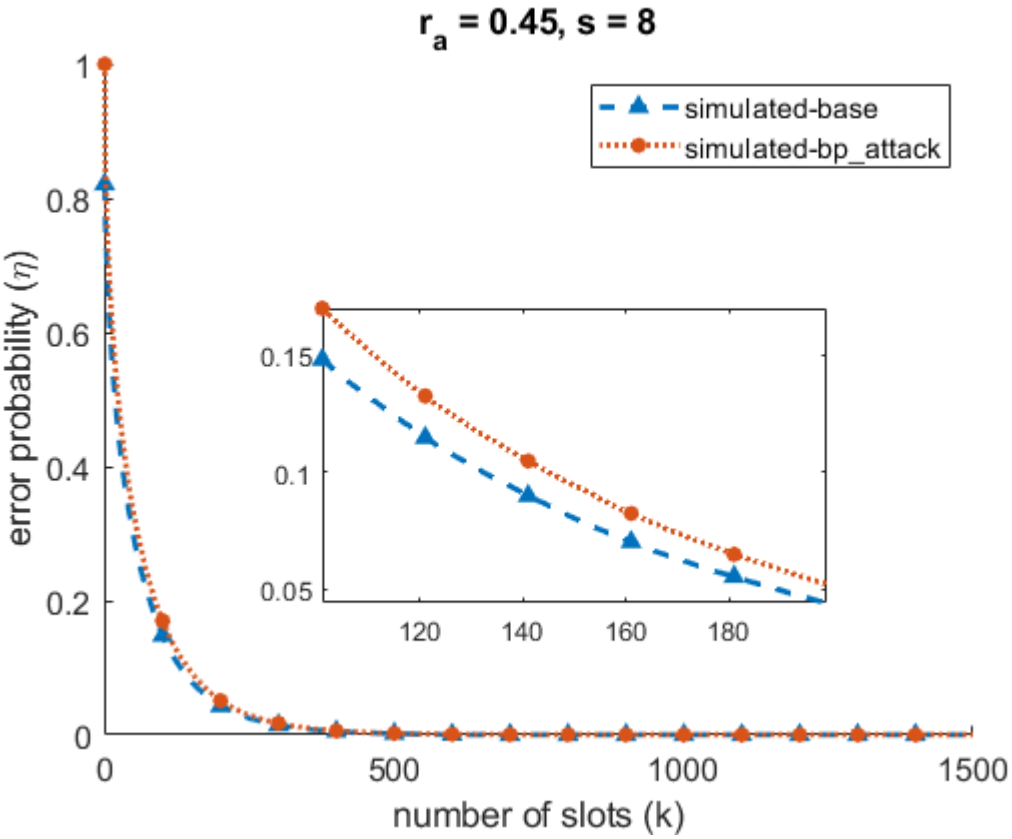
  If $w > u$, then N gains utility t. If $u \geq w$, then N gains utility $(w + t - u)$.



(a) Optimal expected gain given v, t, $r_a = 0.45$



(b) Expected gain given c, v, $r_a = 0.45$, $t = 0$



(c) Effectiveness of the borrow power attack

Figure 4.9: Borrow Power Attack – Simulation Analysis

N's expected gain: $c4_h = \int_v^{v+t} au^{a-1} \left(\int_u^1 tcw^{c-1}\,\mathrm{d}w + \int_v^u (w+t-u)\,cw^{c-1}\,\mathrm{d}w \right) \mathrm{d}u$

- Case 5: $w + t > u > v + t$

  If $w > u$, then N gains utility t. If $u \geq w$, then N gains utility $(w + t - u)$.

  Adversary gains utility $(u - (v + t))$

  Adversary's expected gain: $c5_a$

  $$= \int_v^{1-t} cw^{c-1} \left(\int_{v+t}^{w+t} (u - v - t)\,au^{a-1}\,\mathrm{d}u \right) \mathrm{d}w + \int_{1-t}^1 cw^{c-1} \left(\int_{v+t}^1 (u - v - t)\,au^{a-1}\,\mathrm{d}u \right) \mathrm{d}w$$

  N's expected gain: $c5_h = \int_{v+t}^1 au^{a-1} \left(\int_{u-t}^u (w + t - u)\,cw^{c-1}\,\mathrm{d}w + \int_u^1 tcw^{c-1}\,\mathrm{d}w \right) \mathrm{d}u$

- Case 6: $u > w + t > v + t$

  Adversary gains utility $w - v$.

  Adversary's expected gain: $c6_a = \int_v^{1-t} cw^{c-1} \left(\int_{w+t}^1 (w - v)\,au^{a-1}\,\mathrm{d}u \right) \mathrm{d}w$

So the expected gain of the adversary is, $f_{eag}(a, c, v, t) = (c5_a + c6_a) - (c1_h + c4_h + c5_h)$
Now, to calculate the optimal c, the adversary will do the following:

1. Substitute $a = \alpha_h - c$ in f_{eag}.
2. Now, find the $c(v, t)$ that maximizes $f_{eag}(c, v, t)$.
3. Substitute v and t, in $c(v, t)$, to get the optimal value of c.

Refer to Fig. 4.9a, to see how the maximum (over c) expected value of f_{eag} changes with v and t, and 4.9b, to see how the value of f_{eag} changes with c and v, when $t = 0$.

As we see in Fig. 4.9c the borrow power attack does not significantly affect the adversary's power to violate finality.

4.9 A Comparative Note on v1, Praos and Algorand

In this section, we compare *QuickSync* with v1, Praos and Algorand.

The first version of the Ouroboros protocol v1, requires synchrony using NTP. Additionally, it assumes that the adversary can not corrupt an honest node in the time span of an epoch, which could last for a few hours or a few days. Thus, it is not secure against FACs, i.e., against a dynamic adversary. This is due to the fact that all the block publishers of an epoch are common knowledge at the start of the epoch.

The next version of the Ouroboros protocol, Praos, provides security against FACs. Praos does not improve upon the performance of Ouroboros v1. Praos achieves liveness and persistence under what

the authors refer to as *semi-synchronicity* (using NTP). However, in Praos, for security, one must set the parameter f appropriately, which implicitly requires prior knowledge of the block propagation delay τ (as well as t_{sl}). The later versions of Ouroboros, *Genesis* and *Crypsinous*, further add features to the Ouroboros protocol, but do not improve upon the performance of Ouroboros v1.

Another popular PoS-based protocol, Algorand [31], is independent of the Ouroboros family of protocols. It too requires strong synchrony (using NTP) to achieve *liveness* and *final consensus*. However, it requires $\frac{2}{3}$ majority of honest nodes as opposed to $\frac{1}{2}$ as required by *QuickSync*, Ouroboros, *Bitcoin* and many others. Due to this strict assumption, we do not consider Algorand to be in the same league and hence, do not present a comparison. Also, the claims of tps achieved in the Algorand paper are based on tightly parameterized simulations, whereas *QuickSync* has been bench-marked using generous margins on real *Bitcoin* network data. We believe that, when using the parameters used by Algorand, *QuickSync* too can achieve similar tps.

Table 4.2 gives the value of $t_f = k \cdot t_{sl}$ for different values of r_a and $1 - \eta$. We have used $\tau = 40$ sec for the comparison. When the adversary has relative stake 10%, it can be easily seen that to have the guarantee of 99.9% on a published block, *QuickSync* needs to wait only for 4 minutes where as v1 needs 10 minutes and Bitcoin needs 50 minutes. It can be seen from Table 4.2, *QuickSync* is about 3 times better than v1 and roughly an order of magnitude better than Bitcoin for the same level of guarantees on finality. In Table 4.1, we compare the throughput, and it is clear that *QuickSync* performs the best in this regard, better than Bitcoin by an order of magnitude.

4.10 Discussion on *QuickSync* Parameters

In this section, we discuss how to set the three parameters relevant to the protocol *QuickSync*. Two parameters are endogenous to the protocol, namely, s and t_{sl}. s is the factor by which the relative stake of a node is scaled to assign it the stake power. In *QuickSync*, s affects the t_f significantly and hence should be chosen to be optimal. We recommend using $s = 8$. As we see from Fig. 4.7a any $s > 4$ should suffice. t_{sl} is the time slot length in units of time. t_{sl} greatly impacts transaction per second (tps), time to finality (t_f), and relative stake of the honest nodes (r_h). Ideally, t_{sl} must be set equal to τ or at least τ. τ is upper bound on the block propagation delay. It depends upon the network and must be estimated. Decreasing t_{sl}, increases tps and reduces t_f which is very favourable. However, this comes at the risk of reducing it below the actual value of τ, causing r_h to drop below half, compromising the security of the protocol. Given the current state of the internet we suggest $t_{sl} = \tau = 40$ sec (from [23, 21]). Once we set t_{sl}, it need not be changed as internet connectivity will only get better.

Another crucial parameter is k, which is exogenous to the protocol. It is an important factor from the perspective of the consumer of the blockchain. k is the number of blocks that a node should wait to confirm a block for a given confidence level $1 - \eta$ against an adversary with relative stake r_a. As the block gets deeper into the blockchain, the confidence of the block's placement in the ledger increases.

Table 4.1: Comparison of tps

	tps
Bitcoin	$\frac{tpb}{avg.timeperblock(sec)} = \frac{2000}{600} = 3.3$
Ouroboros v1	$\frac{tpb \times r^{active}}{t_{sl}} \implies tps < 50 \; \forall \; r^{active} < 1$
QuickSync	$\frac{tpb}{t_{sl}} = 50 \; \forall \; r^{active} > 0$

Table 4.2: Comparison of Time to Finality (in minutes)

BTC: Bitcoin, v1: Ouroboros v1, QS: *QuickSync*

r_a \ $1-\eta$	0.95			0.99			0.995			0.999		
	BTC	v1	QS	BTC	v1	QS	BTC	v1	QS	BTC	v1	QS
0.10	30	4	2	40	6	2	40	8	3	50	10	4
0.15	30	5	2	40	10	4	60	11	4	80	16	6
0.20	50	8	3	70	14	5	80	17	6	110	24	8
0.25	60	13	4	100	23	8	120	27	10	150	37	13
0.30	100	22	8	160	38	13	180	46	15	240	63	21
0.35	170	42	14	270	74	24	310	88	28	410	121	38
0.40	360	105	31	580	183	55	670	217	66	890	296	90
0.45	1370	486	127	2200	831	226	2560	980	268	3400	1327	361
0.46	2110	794	203	3400	1347	355	3960	1586	428	5260	2143	584
0.47	3710	1487	362	5970	2506	632	6950	2946	747	8330	3969	1041
0.48	8030	3588	826	-	5991	1434	-	7028	1680	-	9438	2335

4.11 Conclusion

PoS-based protocols are attracting attention in the literature but may potentially be vulnerable to FACs, as in v1. In this chapter, we proposed a novel PoS-based blockchain protocol, namely, *QuickSync*. To design it, we introduced a block power metric based on a *Sybil attack resistant function* (Eqn 4.1). We showed that *QuickSync* satisfies chain prefix, chain growth, and chain quality properties with appropriate parameters (Theorem 1). We also showed that *QuickSync* is resistant to FACs (Proposition 4). Our analysis showed that *QuickSync* performs better than the Ouroboros protocols for t_f (time to finality; by a factor of 3) and tps (transaction per second). We leave it for future work to explore the application of our BPSM and CSR to build blockchain protocols outside the Ouroboros framework.

We believe the concept of *block power* and chain power metrics are very useful in designing PoS-based blockchain protocols. The possible applications of Sybil attack resistant functions in other scenarios involving Sybil attacks need to be further explored.

Chapter 5

Conclusions and Future Work

In this thesis we address two important challenges: transaction processing in the Bitcoin ecosystem under *transaction-fee-only* model and security and performance of the Ouroboros v1 PoS blockchain protocol. Firstly, we show that in *Bitcoin* under *mature operating conditions* (MOC), miners following FIFO processing take heavy losses as compared to the ones mining *greedily*. Also, this causes *stranded transactions*, which in turn causes the *price of consumption* to rise. These issues culminate in unfairness for both the miners and the users. Thus, we say that Bitcoin, in its current form, is unfair under MOC. We solve these issues of unfairness by proposing a novel protocol, *BitcoinF*, for processing transactions. *BitcoinF* enforces a minimum transaction fee and uses two queues, instead of one to process transactions. We believe *BitcoinF* will lead to a stable ecosystem and hence will be fair to the miners and the users. Secondly, we propose a novel PoS blockchain protocol, *QuickSync* that is secure against fully adaptive corruptions, and achieves slightly better tps, and improves on the t_f by a factor of 3, as compared to Ouroboros v1. The key idea is, we propose a metric called block power, assigned to each block. For resistance against *Sybil attacks*, we present a *Sybil attack resistant function*, which we utilize for the block power metric, with the help of a technique called *histogram matching*.

We believe that further research in these directions would prove to be very useful. The effectiveness of *BitcoinF* can be analyzed in combination with other solutions that target transaction-fee-only model and under different input transaction profiles. *QuickSync* can be designed as an independent protocol without using the Ouroboros framework; the essence of the block power ideology can be used to create a block DAG (directed acyclic graph), or perhaps an *absolute finality* blockchain; and the Sybil attack resistant function itself can be used in various strategic scenarios requiring Sybil resistance.

Chapter 6

Notation

Table 6.1: Chapter 3 Notation

Notation	Description
TFOM	Transaction-fee-only model
MOC	Mature Operating Conditions, i.e., TFOM and standard demand
processing latency	time taken to process a transaction
price of consumption	minimum transaction fee required to have the transaction processed
stranded transactions	transactions that face unreasonably high processing latency
FIFO	First-In-First-Out
FM	Free Market
$blocksize_{max}$	maximum size of the block, in terms of transactions
f_{txn}	transaction fee offered by a transaction
f_{min}	price of consumption as perceived by the users
f_{extra}	extra fee included by the user to incentivize the miner to prioritize the transaction
η	a users aggression level towards paying higher f_{extra}
ϕ	function users use to calculate f_{extra} given η
ψ_λ	distribution that captures the fraction of users having aggression level η
δ	granularity of the miners mining power considered for investigating equilibrium
β	fraction of greedy miners
α	fraction of the block reserved as FIFO section in *BitcoinF*

Table 6.2: Chapter 4 Notation

Notation	Description
BPSM	Block Publisher Selection Mechanism
CSR	Chain Selection Rule
SBP	Selected Block Publisher
PoW, PoS	Proof-of-Work, Proof-of-Stake
v1, Praos	Ouroboros v1, Ouroboros Praos
FAC	Fully Adaptive Corruption
tps	transactions per second
t_f	time to finality
pk_i, msk_i	public key, master secret key of node i
tpb	transactions per block
$len(C)$	length of chain C excluding genesis block
r_h, r_a	relative stake of honest nodes, adversary
NTP	Network Time Protocol
t_{sl}	length of a slot
B_{PG}^{ep}	Pseudo-genesis block of epoch ep
sk_i^l	secret key of node i for slot l
$seed^{ep}$	random seed used in epoch ep
r_i^{ep}	relative stake of node i considered in epoch ep
VRF	Verifiable Random Function
$\sigma_{uro}, \sigma_{proof}$	uniform random output of the VRF, its proof
κ	length of σ_{uro}
η	probability of finality violation with fork starting from a given point
τ	block propagation delay
$P(B), P(C)$	power of block B, chain C
s	scale factor
α	stake power; $\alpha = r \times s$;
B_C^l	block B, part of chain C, published in slot l
C_{csr}	chain selected by CSR; adopted/held by the node
B_i^l	block B, published in slot l by node i
Bh_i^l, Bd_i^l	block header, data of block B_i^l
$S_{view(i,l)}^{Chains}$	set of chains known to node i at the start of slot l
$S_{view(i,l)}^{validChains}$	chains of valid length in $S_{view(i,l)}^{Chains}$
MTR	Merkle Tree Root
r^{active}	fraction of active relative stake
L	lifetime of the protocol in slots
k	common prefix parameter

Publications

Related Publications

- S. Siddiqui, G. Vanahalli, and S. Gujar. Bitcoinf: Achieving fairness for bitcoin in transaction fee only model. In *International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2020*, 2020
- S. Siddiqui and S. Gujar. Quicksync: A quickly synchronizing pos-based blockchain protocol. *arXiv preprint arXiv:2005.03564*, 2020

Other Publications

- A. Jain, S. Siddiqui, and S. Gujar. We might walk together, but i run faster: Network fairness and scalability in blockchains. In U. Endriss, A. Nowé, F. Dignum, and A. Lomuscio, editors, *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, 2021

Bibliography

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

[2] S. Arora, A. Jain, S. Damle, and S. Gujar. Ashwachain: A fast, scalable and strategy-proof committee-based blockchain protocol. In *Workshop on Game Theory in Blockchain at WINE*, volume 2020, 2020.

[3] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930. ACM, 2018.

[4] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Annual International Cryptology Conference*, pages 431–448. Springer, 1999.

[5] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.

[6] bitinfocharts.com. Bitcoin hashrate chart. `https://bitinfocharts.com/comparison/bitcoin-hashrate.html`.

[7] blockchain.com. Average number of transactions per block. `https://www.blockchain.com/charts/n-transactions-per-block?timespan=all&daysAverageString=7`.

[8] blockchain.com. Transaction rate. `https://www.blockchain.com/en/charts/transactions-per-second`.

[9] blockchain.com. Mempool transaction count. `https://www.blockchain.com/charts/`, 2019. Accessed: 15-November-2019.

[10] V. Buterin and V. Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[11] Cardano. Introduction to cardano. `https://cardanodocs.com/introduction/`.

[12] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.

[13] I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.

[14] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui. Localcoin: An ad-hoc payment scheme for areas with high connectivity: Poster. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 365–366, 2016.

[15] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui. Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 442–450. IEEE, 2018.

[16] D. Chatzopoulos, A. Jain, S. Gujar, B. Faltings, and P. Hui. Towards mobile distributed ledgers. *arXiv preprint arXiv:2101.04825*, 2021.

[17] cs.cornell.edu. Bernstein. `https://www.cs.cornell.edu/~sridharan/concentration.pdf`.

[18] S. Damle, S. Gujar, and M. H. Moti. Fasten: Fair and secure distributed voting using smart contracts. *arXiv preprint arXiv:2102.10594*, 2021.

[19] S. Damle, M. H. Moti, P. Chandra, and S. Gujar. Designing refund bonus schemes for provision point mechanism in civic crowdfunding. *arXiv preprint arXiv:1810.11695*, 2018.

[20] 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.

[21] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[22] digiconomist.net. Bitcoin energy consumption index. `https://digiconomist.net/bitcoin-energy-consumption`.

[23] DSN. Bitcoin network monitor - dsn research group, kastel @ kit. `https://dsn.tm.kit.edu/bitcoin/#propagation`.

[24] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 327–344, 2019.

[25] earn.com. Bitcoin fees for transactions. https://bitcoinfees.earn.com/, 2019. Accessed: 15-November-2019.

[26] D. Easley, M. O'Hara, and S. Basu. From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics*, 2019.

[27] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[28] 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.

[29] P. Gaži, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 85–92. IEEE, 2018.

[30] P. Gazi, A. Kiayias, and D. Zindros. Proof-of-stake sidechains. In *IEEE Symposium on Security & Privacy*, 2019.

[31] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[32] N. Houy. The economics of bitcoin transaction fees. *GATE WP*, 1407, 2014.

[33] G. Huberman, J. Leshno, and C. C. Moallemi. An economic analysis of the bitcoin payment system. *Columbia Business School Research Paper*, 17(92), 2019.

[34] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *Annual International Cryptology Conference*, pages 332–354. Springer, 2001.

[35] A. Jain and S. Gujar. Block rewards, not transaction fees keep miners faithful in blockchain protocols. In *Workshop on Game Theory in Blockchain at WINE*, 2020.

[36] A. Jain, S. Siddiqui, and S. Gujar. We might walk together, but i run faster: Network fairness and scalability in blockchains. In U. Endriss, A. Nowé, F. Dignum, and A. Lomuscio, editors, *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, 2021.

[37] S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and t-pake in the password-only model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 233–253. Springer, 2014.

[38] S. Kasahara and J. Kawahara. Effect of bitcoin fee on transaction-confirmation process. *arXiv preprint arXiv:1604.00103*, 2016.

[39] T. Kerber, M. Kohlweiss, A. Kiayias, and V. Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *Ouroboros Crypsinous: Privacy-Preserving Proof-of-Stake*, page 0. IEEE, 2018.

[40] A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. *IACR Cryptology ePrint Archive*, 2015:1019, 2015.

[41] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

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

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

[44] D. Koops. Predicting the confirmation time of bitcoin transactions. *arXiv preprint arXiv:1809.10596*, 2018.

[45] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013.

[46] Y. Kwon, H. Kim, J. Shin, and Y. Kim. Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash? *arXiv preprint arXiv:1902.11064*, 2019.

[47] D. Larimer. Transactions as proof-of-stake. *Nov-2013*, 2013.

[48] D. Larimer. Delegated proof-of-stake consensus, 2018.

[49] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. Citeseer, 2015.

[50] J. Li, Y. Yuan, S. Wang, and F.-Y. Wang. Transaction queuing game in bitcoin blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 114–119. IEEE, 2018.

[51] B. Magazine. What is the bitcoin block size limit? `https://bitcoinmagazine.com/guides/what-is-the-bitcoin-block-size-limit`, 2019. Accessed: 15-November-2019.

[52] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 549–566. IEEE, 2018.

[53] M. Möser and R. Böhme. Trends, tips, tolls: A longitudinal study of bitcoin transaction fees. In *International Conference on Financial Cryptography and Data Security*, pages 19–33. Springer, 2015.

[54] M. H. Moti, D. Chatzopoulos, P. Hui, B. Faltings, and S. Gujar. Orthos: A trustworthy ai framework for data acquisition. In *International Workshop on Engineering Multi-Agent Systems*, pages 100–118. Springer, 2020.

[55] S. Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. *Working Paper*, 2008.

[56] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[57] P. R. Rizun. A transaction fee market exists without a block size limit. *Block Size Limit Debate Working Paper*, 2015.

[58] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[59] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

[60] S. Siddiqui and S. Gujar. Quicksync: A quickly synchronizing pos-based blockchain protocol. *arXiv preprint arXiv:2005.03564*, 2020.

[61] S. Siddiqui, G. Vanahalli, and S. Gujar. Bitcoinf: Achieving fairness for bitcoin in transaction fee only model. In *International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2020*, 2020.

[62] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013(881), 2013.

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

[64] A. Tomescu and S. Devadas. Catena: Efficient non-equivocation via bitcoin. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 393–409. IEEE, 2017.

[65] wikipedia.org. Histogram matching, Feb 2019.

[66] M. Yung. The" mobile adversary" paradigm in distributed computation and systems. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 171–172, 2015.

[67] R. Zhang and B. Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy (SP). IEEE*, 2019.

[68] J. Zou, S. Gujar, and D. Parkes. Tolerable manipulability in dynamic assignment without money. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.