Doctoral Thesis Proposal

Security, use and scalability of blockchain systems

Lucianna Carvalhaes Kiffer
Khoury College of Computer Sciences
Northeastern University
lkiffer@ccs.neu.edu

December 11, 2020

Abstract

The celebrated Nakamoto consensus protocol [1] introduced in 2008 has ushered in several new consensus applications, most popularly cryptocurrencies like Bitcoin and Ethereum. There has since been a spark in this new area of study in both academia and industry, including many new systems which are now part of a multi-billion dollar industry [2]. At the heart, these protocols implement a public and immutable record of transactions known as the blockchain.

Development of blockchain systems has been largely led by software development in an ad-hoc manner in which these systems are being built to be functional fast and often lack a much needed formal review. My proposed plan for my thesis is to aid in our understanding of these decentralized computing platforms, their current limitations, how to surpass those limitations, and the properties these systems have and should possess. My research so far has moved towards this goal in several ways: several measurement studies of Ethereum, the second largest cryptocurrency by market cap [2], focusing on several aspects of the network and its history. The goal of these measurement studies is to not only better understand how Ethereum is currently being used, but to also formulate the kinds of systems that can be built on top of the Ethereum architecture or systems like it and the security risks their design possess.

My work also includes a theoretical analysis of a fundamental property of blockchain protocols, called consistency, which is a guarantee that all honest participants of the network output the same blockchain. We use our framework to experimentally analyze the consensus bounds of three blockchain protocols and the impact of attacks on these protocols.

Finally, part of this thesis involves work on developing new blockchain protocols essential in the scalability and incentive mechanisms of these systems. This includes creating and formally proving the security of a proof mechanism that allows for better light clients to participate in transaction verification from light weight devices without having to store the gigabytes or terabytes of full blockchains. The work on miner incentives includes introducing a new family of mining reward functions and an analysis of equilibrium properties of these functions.

The goal of my thesis is to better model and reason about how blockchain protocols are being used and evolve, how they can scale, and to help build a foundation for the rigorous analysis needed in this new and fast evolving area. The results of this thesis can be used to instruct next-generation blockchain system design and analysis both with the tools we use to analyze protocols as well as by understand how current systems are operating.

Contents

1 Introduction

The use of blockchains—for example, in systems like Bitcoin and Ethereum—is an increasingly popular approach to building distributed systems. While currently most well-know as the basis for cryptocurrencies, blockchains have also been proposed to serve as the basis for domain registries [3], medical records [4], and asset registries [5], just to name a few. Blockchains are, at their core, a mechanism for storing state across a network of machines without any centralized trust. Combined with proof-of-work-based limits to generating blocks (called *mining*), blockchains can provide a tamper-proof way to store information.

My proposed thesis is a broad examination of the security of probabilistic blockchain protocols like Bitcoin's proof-of-work protocol. I propose a study of the foundations of these decentralized computing platforms, assess their current limitations, and design methods to surpass them. My research so far has moved towards this goal in several ways including analytical evaluations, measurement studies and building new protocols. Below is a summary of the work presented in this proposal.

Markov model for blockchain protocols There are several properties blockchain protocols should satisfy, arguably the most important of which is consistency which is a guarantee that all honest participants of the network output the same blockchain, except for some constant number of recent blocks. Our research has included a study which focused on three blockchain constructions (Bitcoin's longest-chain, the heaviest sub-tree, and longest clique protocols) whose mining can be modeled using a random oracle (used to model proof-of-work or other proof-of-X resource probabilistic protocols) with miners having independent probabilities of succeeding in each round. We devised a framework to derive consensus bounds on these three protocols and the impact of attacks on them. I presented this work at CCS'18 [6].

Ethereum measurements To better understand the questions surrounding the security of blockchain protocols our research has included analyzing the Ethereum network, the second largest cryptocurrency by market cap, through measurement projects. In a HotNets'17 paper [7], we analyzed the impact of a large-scale network partition that took place in Ethereum in 2016, The Ethereum Classic Fork. In another work published at IMC'18 [8], we studied the ecosystem of Ethereum smart contracts: what kinds are being built and how they inter-relate. The goal of these measurement studies has been not only to understand how Ethereum is being used, but also to formulate the kinds of systems that can be built on Ethereum-style architectures, and the security risks their designs possess.

As shown in our CCS paper, the ability of a blockchain system to arrive at consensus is directly linked to the propagation delay of the network, i.e. the time it takes for the miners in the network to hear about a block. In Ethereum, fast propagation is especially crucial as the expected interval between blocks is around 15 seconds. A miner's individual delay in the network also impacts how often they are able to win the block race (i.e. be rewarded for their work). Our recent work has been a 7 month measurement of network level communication in Ethereum's P2P network. By running a node in the network with a large number of peers and over several months and in different locations throughout the world we've been able to get a unique view of the network. Our results expose an incredible amount of churn in peers, and a surprisingly small number of peers who actually participate in propagating new blocks. We also find that a node's location has a significant impact on when it hears about blocks, and that this behavior has changed over time. Quantifying churn and network

latency is a step in evaluating how the network's behavior matches its security assumptions and how it may scale.

Super-light client for cryptocurrencies When it comes to the future of blockchain protocols and their scalability, a main concern is the growing size of the blockchain. For clients to be able to interact with the blockchain in a trust-less manner, they generally must hold and verify a copy of the blockchain. This limits light-weight client devices from participating in the blockchain ecosystem without compromising on security assumptions. As an intern at Visa Research, I helped create and formally prove the security of a protocol called FlyClient, a probabilistic mechanism for lightweight client devices to participate in transaction verification with similar trust assumptions as the underlying blockchain protocol but without needing access to the whole gigabytes worth of the blockchain (just a logarithmic amount of it). Flyclient is a proof protocol that creates succinct, non-interactive probabilistic proofs that a certain block belongs to the chain with most PoW (or proof-of-X resource), i.e. the chain agreed upon by the network. Flyclient is currently being implemented and considered by several blockchain projects including Beam, Mimblewimble, and ZCash and was presented at SP'20 [9].

Designing a mining reward function Another factor in how a cryptocurrency scales is how miners in the system are rewarded for creating blocks (i.e. securing the system). There are currently two models for the mining reward: Bitcoin's model of halving the block reward approximately every 3 years, and Ethereum's model of keeping the block reward set to 5 ETH for the rest of time (or until a fork changes this). Both variants have their benefits and shortcomings. The block reward going down over time reduces the amount of newly minted coins coming into the system over time which is more favorable for the growth in value of the coin. Bitcoin's halvings, however, are a chaotic, periodic event which results in miners leaving the system till the price of the coin makes up for the miner's loss in revenue. We propose a mining reward function that is pegged to the hash rate of the system, so as new miners join the system the reward goes down. We are exploring equilibria properties of this function.

Outline The remainder of my proposal is structure as follows: I begin by providing background and introducing definitions in § 2. Then I present our work analysing the security of blockchain protocols in § 3. In § 4 I describe the main results of our measurement studies of the Ethereum network. Lastly, in § 5 I present our new protocols FlyClient and HaPPY-Mine, and then conclude by describing my research plan in § 6.

2 Background and Definitions

We now describe the basics of blockchain protocols and, in greater detail, the Ethereum protocol as it has been the focus of our measurement studies.

2.1 Blockchain Basics

Overview Most cryptocurrencies, including Bitcoin and Ethereum, maintain an append-only ledger, known as a *blockchain*, which stores a sequence of blocks of transactions chained together via cryp-

tographic hashes. These blocks are created and appended to the blockchain via a mining process, where the participating nodes, known as *miners*, compete to become the next block proposer, usually by solving a computationally-intensive puzzle, known as a *proof of work (PoW)* [10], with sufficient difficulty. To ensure distributed agreement in the case where multiple blockchains exist, participants in the network choose to believe the chain that represents the most work (typically by picking the longest chain). Through a gossip protocol initiated by the block proposer, every miner receives each block including a PoW solution and appends the block to their local copies of the blockchain if the solution is valid.

Miners and users in these systems communicate through a P2P network. Users initiate a transaction which is gossiped through the network until it reaches miners who check its validity(e.g. the $ has not yet been spent and signatures are correct) and include it in the current block they are working on. Users include a transaction fee along with their transaction to incentivize miners to include it in a block. The miner whose block becomes part of the chain collects the transaction fees of the block as well as a block reward to offset the cost of computing the PoW(hardware and operational costs). Because of the high variance in winning a block, miners often come together to form *mining pools* where block rewards are split among the participants.

Forks Because of the probabilistic nature of mining and the delay in the network, it often happens that two or more valid blocks are mined within a short interval of each other by two different miners, contending for the same position in the chain. In the work presented in this thesis [6, 9] we generally consider an adversary's goal as being to attack consensus (players in the network having an inconsistent view of the chain) by forking the chain in some way. To agree on the same chain consistently with other miners, each miner downloads and verifies all blocks in every chain and picks and follows the one with the largest total difficulty. Using this *most difficult chain principle*, it is shown that, in the long run, the network will agree on a single chain [11, 12] as well as our work [6], known as the *honest (valid) chain*. Participants will switch over to the longer chain and abandon the shorter one, effectively removing the fork.

Systems that use blockchains are often under active development and occasionally need to update the software that nodes run. If these changes are backwards-compatible, they are often termed *soft forks* since the changes don't cause the blockchain to physically fork; in the cases where the change is not backwards compatible (e.g., if the network messages or blockchain data change), they are termed *hard forks* as the two protocols won't produce compatible blocks. These changes are often rolled out as new software versions. In the case of hard forks, it is typical for the developers to release the updated software but announce a specific block number (or time) at which the change will actually be activated; doing so provides users with sufficient time to upgrade their software before the change happens. When a significant part of the community does not adopt the hard fork, a persistent fork may occur. In this thesis we study such an example in Ethereum [7] which results the two chains of the fork becoming independent cryptocurrencies (Ethereum and Ethereum Classic) which both still exist today. There are also examples when protocol changes that are known to be controversial are released with the intention to create a competing currency from a fork. Such examples of this in Bitcoin are Bitcoin Cash [13] and Bitcoin Gold [14] which both still exist today.

Modeling blockchain protocols Theoretically analyzing blockchain protocols requires abstracting away a lot of the complexity present in these active systems. When analyzing consensus and

incentives in the system, the players are the miners (who create blocks and maintain the blockchain) and the users (who input transactions into the system and under certain models may want a full or partial view of the blockchain). When considering an adversary in the system, we generally consider one who is computationally bounded by some fraction of the mining power. To abstract away the P2P network we may give the adversary partial control(e.g. imposing delays and reordering of messages), while maintaining some guarantees that messages(e.g. blocks and transactions) eventually get delivered. Finally, mining success are modelled as a random process with expectations of success. Considering worst-case adversarial mining power and control of the network allows us to make strong statements on the security of the protocols based on the cost on the adversary to attack such protocols.

2.2 Ethereum Specifics

Ethereum is a blockchain-based distributed system, much in the spirit of related systems such as Bitcoin. However, it has a number of unique features that make it distinct including the use of smart contracts, a shorter block interval of 15 seconds (compared to 10 minutes in Bitcoin) and a general purpose network peer discovery protocol.

Smart Contracts Unlike other blockchain-based systems that primarily serve as a virtual currency, Ethereum serves both as a virtual currency *and* a distributed virtual machine. Users can upload *code* to Ethereum—called *smart contracts*—that are run in a deterministic fashion by all participants; each contract has its own memory state and currency balance. The creating user and others can later invoke (or *call*) these contracts, in response to which contracts can transfer funds or call and even create other contracts. These calls are made via transactions where the user who sends the transactions also includes fees paying each operation the code executes, and each byte of memory it uses. These fees are again collected by the miner who includes the transaction, but everyone on the network that is verifying the blockchain must also run the computation of each transaction. Ethereum blocks include transactions and a hash commitment to all contract and account states after all transactions in the block are run. Though a transaction may trigger many calls within the contract, those calls are not explicitly logged in the blockchain, just the final state.

Uncles and Prunes Like with other blockchain protocols, the probabilistic nature of mining means there are times when two miners may solve the PoW for a block at roughly the same time. Ethereum's shorter block time introduces more competing blocks and the need to reward miners for work that does not make it to the mainchain. To still reward mining related to these discarded blocks, miners can also choose to include these valid, but non-`mainchain` blocks as `uncles` in the blocks they mine. A miner includes the hash of the discarded block in a special field of the block only if the parent of the `uncle` (the block the `uncle` points to) is a block in their own chain up to six blocks prior. Both the miner and the miner of the `uncle` receive an additional, smaller reward. Finally, some valid blocks may be mined, but never end up in the `mainchain` or become `uncles` (e.g. if the announcement of the block is significantly delayed), we refer to such blocks as `prunes`. In blockchain protocols that don't include `uncles`, all non-`mainchain` blocks are `prunes`. Since `prunes` are not recorded on the `mainchain`, the only way to know about them is to be listening on the network and recording all block messages.

Network The Ethereum system is made up of three layers: the *application layer* that contains the blockchain, the *Ethereum layer* that contains peers exchanging information about blocks and transactions, and the *peer-to-peer (P2P) layer* that allows nodes to find others and establish connections.

The P2P layer is divided into two components: a *discovery protocol* that allows nodes to find each other, and *DevP2P* that nodes use to communicate. The discovery protocol outlines how a client learn about other nodes in the network from their peers and is based on the *Kademlia* distributed hash table protocol [15]. The discovery protocol is a general purpose protocol for multiple other networks related to Ethereum, as such, an Ethereum client often hears about nodes not running the Ethereum protocol. The DevP2P protocol runs in parallel to the Discovery protocol and is responsible for establishing sessions with other nodes, sending and receiving messages between peers and managing the actual higher-level protocol being run. Importantly, the DevP2P protocol checks if a remote node is running the same application-layer protocol (e.g. `eth` for the Ethereum Wire Protocol), that they support each others' protocol version, and that they agree on the blockchain (i.e., the genesis blocks and any forks). If all conditions hold, the nodes become *peers* and can exchange messages at the Ethereum layer. The Ethereum Wire Protocol [16] is the application-layer protocol for propagating transactions and blocks, and for requesting block and state data so new clients can sync to the existing state.

Data Collection In our measurement studies of the Ethereum network we are interested both on data stored on the blockchain as well as network-level messages. By design these system run a publicly available blockchain with open source client code over an open network that anyone can join. Thus, to collect our data we instrument a version of the client which connects to the network and downloads the full blockchain, re-runs all code logging information of interest, and logs all network-level messages received. This way we have access to all `mainchain` blocks and their transactions, all calls made by these transactions, the `uncles` and `prunes` from when our clients are running, as well as timing for when our peers are online and the messages they send us.

3 Analyzing Consensus

In 2008, Nakamoto [1] proposed the idea of using *proofs of work* to implement a public, immutable and ordered ledger of records suitable for applications such as cryptocurrencies. While standard consensus/Byzantine agreement mechanisms could be used to achieve such an immutable ordered sequence of records, the amazing aspect of Nakamoto's protocol is that it functions in a fully permissionless setting *and* works as long as more than half of the computing power in the network follows the protocol. In contrast, prior work on Byzantine agreement showed strong lower-bounds in fixed-party settings when even just one-third of the participants were adversarial. It is thus remarkable that the honest parties using Nakamoto can reach agreement on a sequence of blocks. This property has been *strongly* validated by the Bitcoin network for over a decade of operation during which the participation has grown by 12 *orders of magnitude* from millions of hashes/second to million trillions of hashes/second!

To understand this phenomena, the original Nakamoto paper provided the first intuitive analysis as to how the protocol achieved consensus. Specifically, the paper shows that if an honest participant adds a block B to the chain and then waits for k more blocks to be added, the probability that an attacker (with less than 50% of the computational power) can build an alternative chain that

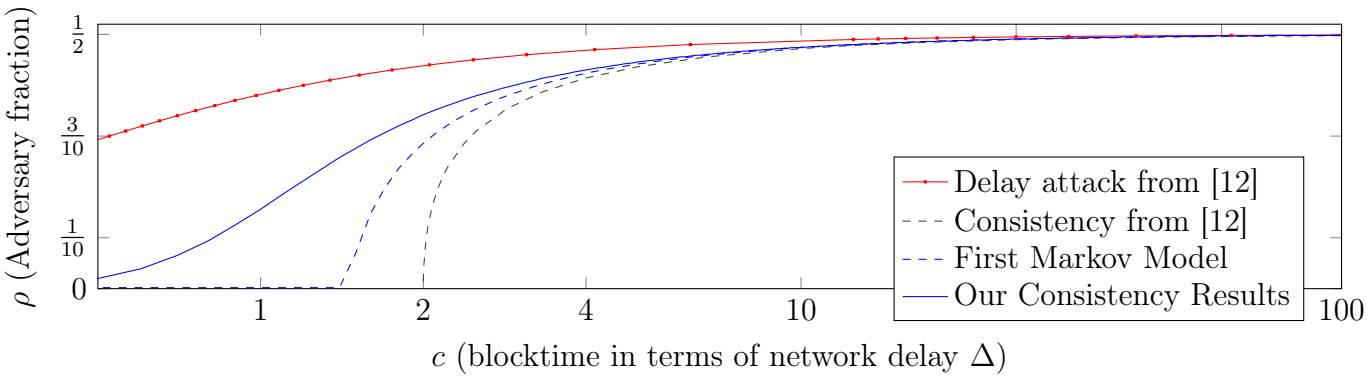

Figure 1: Replication of [12, Fig. 1], "Minimum percentage of computing power an adversary must hold in order to break consistency" using same parameters $n = 10^5$ and $\Delta = 10^{13}$, $p = \frac{1}{c\Delta}$, but illustrating the new bound from our paper [6].

does not include B drops exponentially with k. While intuitive, this analysis unfortunately does not consider other attacks, and thus does not fully establish the consensus property for the protocol. For example, the analysis does not consider an adversary that attempts to introduce small disagreements between honest miners so as to split their computational power among several "forks".

Garay, Kiayas and Leonardos [11] provided the first formal modeling of Nakamoto consensus and proved that the protocol achieved a *common prefix-property*. Their analysis only considered a "static setting in which the participants operate in a synchronous communication network in the presence of an adversary that controls a subset of the players." In particular, players were either honest or adversarial throughout the protocol, and the network model ensured delivery of messages in the next round. Assuming a synchronous network is a very strong, possibly unrealistic assumption; indeed, Nakamoto's protocol is explicitly designed to work in a network *with message delays* such as the public Internet. Furthermore, the notion of *common-prefix* is not strong enough because it does not preclude the chain from alternating between two different versions on even and odd rounds.

To address these issues, Pass, Seeman, and Shelat [12] provide a different formal model for studying the properties of Nakamoto's protocol. In particular, they introduce an idealized model for the protocol execution that can capture adaptive corruptions (and un-corruptions) of parties and a *partially synchronous* network in which the adversary can adaptively and individually delay messages up to some delay limit Δ. They also introduce a stronger notion of consistency and then proceed to show how Nakamoto can achieve this notion when the hardness of the proof of work p is set with an appropriate relation to Δ and the number of participants n. Their analysis is precise enough to make concrete claims about the relationship between p, Δ, and n for which consistency holds and also for which a simple delay attack can violate consistency.

In subsequent work in 2017, Garay, Kiayas and Leonardos [17] studied aspects of how the hardness p is adjusted as more players join the protocol during *epochs* in the Nakamoto consensus protocol and how this epoch must be sufficiently large to avoid certain attacks. Techniques from this paper were also used to update [11]; in particular, the updated version of the latter extends incorporates the definition of consistency used by Pass, Seeman and Shelat and extends the previous analysis to the partially synchronous model but is not precise enough to make concrete claims about the parameters.

3.1 Contributions

In our work [6], we show how Markov chains can be used to model blockchain protocols to both simplify the analysis of blockchain protocols and attacks and to make precise claims about parameter relationships. To introduce our method, we show how to replicate the analysis of Nakamoto's protocol done by Pass et al. [12] using a Markov model. We validate our method by recovering essentially the same bound.

By inspecting this graphical Markov model, however, we discovered cases in which the counting of events in [12] under-counts a special event. Based on this insight, we show how our Markov model *exactly* counts so called "convergence opportunities" and thus leads to more accurate bounds on consistency. To illustrate this new result, Fig. 1 replicates a graph from [12] showing a relationship between proof of work hardness and adversary control. Against the original analysis (- - -) and attack (–■–), our new result from the Markov model (——) shows higher resilience at lower parameters where the previous analyses provided no lower bounds.

To further illustrate our technique, we introduce Cliquechain, a specific example of the Chainweb protocol [18] for which we show the same consistency lower-bound as Nakamoto's protocol. Chainweb proposes a blockchain protocol that creates a braid of various parallel chains. The main idea is that at each level the chains reference each other according to some base graph, and thus in order to replace one block in any chain, you must also replace the blocks in the parallel chains that reference it. The protocol claims to be able to handle 10K transactions per second over hundreds or thousands of parallel chains, but the analysis in the paper only considers variations of the 50% attack. Our analysis on consistency does not support this 10k claim. Bitcoin, running Nakamoto's protocol with network delay of about 10 seconds, can handle approximately 7 txns/sec, we show that the Cliquechain protocol with any number of chains is bounded by the same throughput as Nakamoto's protocol for the same consistency guarantee.

We extend our techniques to establish that the protocol GHOST [19] also has the same consistency lower bounds. GHOST's main claim is also that it can handle higher transaction rates than Nakamoto's protocol while being resilient to 50% attacks.

For each protocol we also analyze variants of attacks using our technique and provide probability distributions for how long the attacks last. We use this attack model to answer very pragmatic questions about blockchains: For example "how long should one wait before confirming a transaction?" While folklore holds that one should ignore (i.e., wait for) the last 6 blocks, we provide a more precise answer to this question by modeling an attack in which the goal of the adversary is to "undo" a recently confirmed transaction. We show, for example, that 6 is a surprisingly low default for chains like Ethereum which use more aggressive network parameters.

Since this delay attack is not successful on GHOST, we present another attack, the 'Balancing Attack', with a Markov chain which captures a simplified version of the attack on GHOST. We use this model to capture a lower bound on transaction confirmation time for the GHOST protocol, along with simulations of the full attack on GHOST.

3.2 Related Work

Our work is not the first to employ Markov-based analysis of blockchains. Indeed, Eyal and Sirer [20] develop a Markov model to analyze the success of the selfish mining attack on Nakamoto's protocol. Our work, however, is the first to use Markov-based models to analyze consistency against any adver-

sary, and provides a general framework to analyze specific attacks on various blockchain protocols. Previous studies on consistency [11, 12, 17] have advocated using Markov methods, but considered them too complicated to analyze.

Chainweb and GHOST are two examples of a new class of blockchain protocols which consider DAG-based chains instead of a linear chain. Other such examples include the inclusive protocol of Lewenberg, Sompolinsky and Zohar [21], SPECTRE, another protocol from the authors of the inclusive protocol [22], and PHANTOM [23]. However, none of these protocols give a formal argument for the consistency properties put forth in either [11,12], but instead mostly rule out specific attacks.

In [24] and [25], Kiayias et al. analyze both Nakatomoto's protocol and GHOST in a synchronous setting. They show an attack on the growth rate of the main-chain to delay transaction confirmation time in this model of both GHOST and Nakamoto's protocol. Their results show that the attack produces greater delays in the GHOST protocol. They extend their analysis of Nakamoto's protocol to the partially synchronous model in a later version of [11]. In our analysis of GHOST we consider a stronger adversary that can delay honest messages up to the network delay, and focus on consistency attacks (instead of chain growth attacks) which also delay confirmation time. Sleepy [26] proposes a new blockchain in a model with a CRS and a PKI and participants who sometimes become inactive; they show how to replace a proof of work with another rate-limiting mechanism. We believe our techniques directly apply because they apply the same counting as [12] (see, e.g. Lemma 2 in [26]). The Algorand schemes [27,28] construct a blockchain from improved Byzantine agreement protocols; as far as we can tell, they require a $2/3$ fraction of honest users and thus rely on different techniques for proving consistency.

The choice of network delay in our analysis is supported by measurements of real delays in active blockchain systems, see below Section 4.3. Apostolaki et al. show how an ISP can partition the Bitcoin network and delay messages [29] thus justifying our choice to allow the adversary the power to rearrange and delay messages between players.

3.3 Future Work

In this work we present a Markov model that is able to better bound the consistency guarantees of the longest chain protocol of Bitcoin and Ethereum. We extended our model to prove bounds for two other blockchain protocols with DAG (distributed acyclic graph) structures rather than a linear chain. A natural extension of this would be to extend the model to a larger class of DAG-based blockchain protocols. Additionally, our analysis relies on counting special convergence events which bear some similarities with the analyses of some Proof of Stake protocols including Ouroboros [30] and the Sleepy model of consensus [26]. Another avenue for future work is to use this style of Markov analysis for non PoW blockchain protocols.

4 Measuring a Blockchain System

Ethereum [31] is a novel cryptocurrency that uses a blockchain not only to store a record of transactions, but also to store user-generated programs called *smart contracts* and a history of calls made to those contracts. Today, Ethereum is the second-most-valuable cryptocurrency behind Bitcoin, with a market capitalization of over $26B [2]. Given its complexity, it may be unsurprising that Ethereum has a number of differences from Bitcoin. Compared to Bitcoin [1], Ethereum is significantly more

expressive and can be used to implement decentralized voting protocols, financial contracts, and crowdfunding programs.

Ethereum is based on a general purpose peer-to-peer (P2P) overlay responsible for discovering other nodes and maintaining connectivity. This P2P layer can be used by higher-level protocols other than Ethereum (in fact, we find this is often the case). The Ethereum Network layer sits between the P2P layer and the overlying application layer (the blockchain itself), and is responsible for choosing peers, disseminating new blocks, and reaching network-wide confirmation of transactions. Ethereum's 15 seconds(on average) block interval is dramatically shorter than Bitcoin's 10 *minutes* (on average). This significantly reduced target block mining interval opens the door to much faster "network confirmation" of accepted transactions. As such, Ethereum represents an interesting case study of measuring a blockchain system. Our measurement work, summarized below, has focused on several aspects of the Ethereum network including a network partition event that has persisted to this day as two separate systems, the study of the smart contract ecosystem of Ethereum, and a longitudinal study of the P2P network used to propagate new blocks and transactions. These studies help us answer the following questions (1) what happens when blockchain system fork? (2) How is Ethereum being used? and (3) How is the Ethereum network operating?

4.1 What happens when blockchains fork?

The inherent decentralization of blockchain systems poses a challenge in how network partitions are handled. A network partition here means nodes can no longer communicate due to a portion of the nodes adopting a new protocol; this is in contrast to a "traditional" network partition where nodes are unable to send messages to each other at all. In traditional distributed systems, network partitions are generally a result of connectivity failures or software misconfigurations—i.e., are unintentional—and systems are typically designed to recover from such instances in as automatic a fashion as possible. Blockchain-based systems, however, rely on purposeful "hard forks" to roll out protocol changes in a decentralized manner. If not all users agree with proposed changes, or not all users upgrade to the new software, a persistent network partition can occur between the nodes that support the newer version of the protocol and those that do not. In this case, there will effectively be two separate systems running in parallel, each with its own state and history. In fact, such a persistent fork occurred [13] in Bitcoin in the summer of 2017, meaning each user who existed before the fork now has two "copies" of their wallet (one in the original Bitcoin, another in the new "Bitcoin Cash") that they can spend independently.[1]

Ethereum is of particular interest when it comes to forks because in July 2016 the system forked into two partitions, both of which still exist today; these two partitions represent two versions of the same currency and are dubbed *Ethereum* (ETH) and *Ethereum Classic* (ETC). In our paper [7], we closely examine the dynamics around the ETH/ETC fork to better understand the behavior of the participants, and the impact a fork has on the security and market value of the overall system. Overall, we make the following observations:

1. Forks can lead to drastic, rapid partitions: ETC experienced a sudden loss of roughly 90% of the nodes in its network immediately after the fork.

[1] In March 2013, Bitcoin also experienced a short-lived fork due to inconsistent protocol versions that lasted approximately four hours before developers intervened [32].

2. Stabilization after forks can take days to occur: It took two days for ETC to resume producing blocks at the target rate; an influx of nodes re-joined ETC over the subsequent two weeks.

3. Forks can persist, with divergent behavior afterwards: ETC's mining power has held constant since the fork, while ETH's has increased tremendously.

4. The "market" between the two networks appears to be operating efficiently, with the expected return (in USD) on mining being almost identical between the two.

5. The fork unintentionally introduced a security vulnerability wherein attackers can rebroadcast transactions into the other network; this continues to this day, and we quantify this behavior.

6. Pool-based mining behavior in ETC has slowly reached a distribution similar to that in ETH.

As the 2017 Bitcoin fork [13] suggests, such network partitions are likely to be a fact of life with digital currencies based on public ledgers. Unfortunately, neither Ethereum nor Bitcoin were designed to operate under a persistent fork. We believe our study to be an important first step towards better anticipating and handling the unintended consequences of persistent forks.

4.2 How is Ethereum being used?

Smart contracts expand the functionality and usability of blockchain-based cryptocurrencies; users have developed contracts to implement voting protocols, funding programs, gambling, and many more. As a result, various other cryptocurrencies are incorporating smart contracts. Smart contracts also introduce a new layer of complexity and interactivity to the already-diverse ecosystem of cryptocurrencies. At the bottom-most layer, most cryptocurrencies are built on top of a peer-to-peer communication substrate (e.g., the Kademlia [15] distributed hash table in Ethereum), which have received significant study [33]. At the topmost layer, users interact with one another, perform transactions, and make large protocol decisions such as when to fork—these dynamics, as well, have been studied extensively [7]. Smart contracts sit somewhere in the middle: users create them and interact with them (e.g., by contributing to a "go-fund-me" contract), but they also serve as an intermediate layer, as contracts can be built to rely on other contracts.

Although smart contracts are in many ways the essence of Ethereum and other contract-based cryptocurrencies, surprisingly little is known about them empirically. Many open questions remain: How many distinct contracts are there? How long-lived are they? To what extent are users creating wholly new contracts versus copying existing code? How many contracts rely on the availability of a given contract?

To answer these questions and more, our paper [8] initiates the study of Ethereum's *smart contract topology*. Viewing contracts as nodes and contract calls as edges, we are able to measure the importance, connectivity, and central points of failure of Ethereum's smart contracts. To do so, we collect the bytecode of all contracts published to the Ethereum blockchain during its first 5 million blocks (almost three years), and also collect a trace of an instrumented Ethereum virtual machine (`geth`) to log all interactions between users and contracts. We apply the results of this analysis to answer two broad questions:

1. How is Ethereum being used? With the rampant speculation in the cryptocurrency markets (e.g., Bitcoin), one may wonder how this has impacted Ethereum. We find that while Ethereum's market

cap and exchange rate has grown over 1,000-fold during our measurement period, the fraction of activity on Ethereum that involves contracts has remained relatively constant (roughly 1/3 of all transactions are destined to contracts, rather than users). However, we do observe evidence of a number of attacks on the Ethereum platform, often exploiting mis-priced virtual machine operations that were later adjusted.

2. How are contracts being used? Given the high level of interaction with contracts, we examine how these contracts are created and how they interact. Surprisingly, we find that roughly three times more contracts today are created by *other contracts* than are created by users; many of these contracts are sub-currency contracts, or cryptocurries built on top of Ethereum. Additionally, we find that roughly 60% of all contracts that have been created have never been interacted with, suggesting there exists significant amounts of dormant code and currency. Finally, we find extremely high levels of code re-use and code similarity: the 1.2M user-created contracts can be reduced to 5,877 contract "clusters" that have highly-similar bytecode. This high level of code re-use suggests that bugs or vulnerabilities in these contracts could easily impact thousands more; such vulnerabilities have been discovered in the past, and have led to hundreds of millions of dollars in lost value [34, 35].

4.3 How is the Ethereum network operating? (On-going work)

The structure of the Ethereum P2P overlay, including aspects such as peer connectivity and block propagation delay, play an important role in achieving the target performance and correct functioning. Most prior measurement work on Ethereum has focused either primarily on the P2P layer [36, 37] or primarily on the application layer (i.e., the blockchain itself [7, 8, 38–40]), leaving the Ethereum network layer less well-understood.

In this project, we aim to better understand the network structure of Ethereum, focusing on both how the Ethereum network is formed and evolves over time, as well as how the network is used to propagate new blocks, a crucial part of the consensus mechanism. We do this by integrating information from the P2P layer (e.g., which nodes are available, who nodes choose to connect to), information from the application layer (e.g., which blocks are ultimately accepted), along with information from the Ethereum layer (e.g., which peers nodes exchange block information with, and which peers actually provide the most useful information).

We conduct our study by running a customized version of Ethereum's official Go client, `geth`, for over 7 months, allowing us to observe the evolution of the Ethereum network through multiple protocol changes. We also run multiple nodes in a variety of vantage points across the globe for shorter lengths during this time period, allowing us to study both how our peers interact and how the location in the physical Internet affects the peers' experience in the Ethereum network. The results of our analysis can be summarized as follows:

- **Extensive peer churn:** We observe dramatic levels of churn in the Ethereum network, both in terms of the number of unique peers and connection lengths. Churn can run the risk of disconnecting a network or making it difficult to quickly propagate information throughout it, and challenging to estimate the size of the network. We investigate churn in the Ethereum network and find that 68% of the peer IDs we see in our 200-day scraping period are present only on a single day, and 90% of them are present on fewer than 25 days.

- **Miner centralization:** The top 15 miners are responsible for over 90% of mined blocks. We

trace these down, and find that all but one are well-known mining pools. We also note a difference in the efficiency of miners, the top 3 mining pools have a much greater probability of a block they mine being included in the blockchain (propagating faster and "winning" the block race).

- **Most announcers are long-lived:** Comparing all peers to those who are first to announce a block we see that those announcing blocks tend to have longer connections, average and total connection lengths, and are online more days. Almost 70% of peers are seen only one day, but about 40% of our announcing peers are seen only one day. The latter is still a large percent but there is a set churn in peers who are not participating in block propagation.

- **Announcers are diverse:** Focusing on *which* peers tell our nodes about new blocks first, we find that a large number of peers are responsible for announcing a miner's block first to our node. No one peer announces more than 6% of a miner's blocks to us.

- **Quick network-wide propagation:** The speed at which new information spreads throughout the Ethereum network is critical in how quickly transactions can complete. We perform a novel analysis of network propagation times by running nodes in three different vantage points (USA, South Korea, and Germany). Despite the diversity of information sources across the three, we find that the difference in time from when the first learns about a block until when the last learns about it is very small: for instance, less than 100 msec for 85% of all new blocks.

- **Location bias:** Although our vantage points in North America, Asia, and Europe did not experience any unfair disadvantages, we observed bias using additional vantages in locations with fewer peers (South America and Oceania). We observe an immense disparity of which locations hear about blocks first, those two locations being first to hear about blocks <1% of the time.

4.4 Related Work

There has been extensive work towards developing an empirical understanding of various aspects of the cryptocurrency ecosystem. This includes both analyzing data extract-able from the blockchain as well as messages being propagated in the P2P network.

Early work in this space inspected the transactions taking place on Bitcoin's blockchain including transaction patterns [38, 41], properties of repeated subgraphs [42] and hypergraphs [43], and the UTXO set [44]. Work focusing on privacy and anonymity of Bitcoin transactions have looked at transaction history [45], address clustering [46] and mixing services [47] to de-anonymize addresses. Others have focused on transactions related to scams, Ponzi schemes and Ransomware [48–50] and the impact of what kind of data is stored as part of Bitcoin transactions [51]. Anderson et al. [38] analyzed the use of three cryptocurrencies including Ethereum, looking at statistics of the kinds of transactions that were taking place in each as of June 2016. Others have used network statistics to evaluate the constraints of scaling decentralized blockchain protocols [52] and how network traffic can de-anonymize Bitcoin addresses [53].

While existing measurement studies have focused on the Ethereum and other cryptocurrency networks under normal operation or under simulated adversarial attacks, none to our knowledge have looked at how Ethereum responded to the Ethereum Classic large-scale fork. The potential for

network partitions is not unique to blockchains, as traditional distributed systems also have state they wish to keep consistent, and aim to largely handle inconsistencies in as automated a way as possible. There is a long line of work focusing on detecting and preventing inconsistencies [54] as well as building fault-tolerant systems [55, 56]. The distinction between blockchain systems and traditional distributed systems is that the shared state is designed to be tamper proof, even if a significant fraction of the network is malicious.

We explore smart contracts as a unique and growing aspect of the cryptocurrency ecosystem. The most closely related work involves analyses of the kinds of contracts being written. Norvill et al. clustered 998 Ethereum contracts whose source code were available on the block explorer `etherscan.io` at the time of their study [57]. They looked at the frequency of the most common words in the code and clusters based on context triggered piecewise hashes of the bytecode of these contracts. Bartoletti and Pompianu studied 811 Ethereum contracts with available source code [58] and categorized them into 5 categories: financial, notary, game, wallet and library. They did a similar analysis for Bitcoin transactions which use a scripting language and encode some metadata in transactions. Their analysis of the Ethereum contracts included manually inspecting them to identify different design patterns. Compared to both of these studies, we perform our analysis over a much larger scale of contracts. We are not limited to contracts for which there is publicly available code; we measure contract similarity based on n-grams of the decompiled bytecode for all unique bytecodes, allowing us to examine similarity at a much larger scale. Since our study, work on smart contracts in Ethereum have included looking at the prevalence of Ponzi schemes [39] and the ecosystem system of special class of contracts called ERC20 contracts blockchain [40].

The first measurement study of network-level messages in Bitcoin was [59] which measured the block propagation delay and fork-rate (find a median and mean delay of 6.5 and 12.6 seconds from 2013). Other papers that most closely relate to our network measurement work are [36] and [37]. In [36], the authors run a measurement-based comparative study of the Bitcoin and Ethereum P2P networks with a focus on decentralization properties. For Ethereum they look at peer bandwidth, connection latency (to peers and bounds between nodes), and some amount of efficiency of miners through miner distribution of blocks and uncle counts. In [37] they scrape the Ethereum P2P network by connecting to peers just long enough to establish a full DevP2P connection and checking up to the DAO fork (i.e. not a ETC node). They focus their analysis primarily on node client type, "freshness", location/ASes and also connection latency. These two papers ran scrapers collecting quick peer information while we run a long-term full node which is able to collect more temporal node information (i.e. analyze churn in more detail), and connect peer information with the kinds of block data they send us (i.e. block propagation, some miner analysis), as well as capture `prune` blocks which has yet to be observed in Ethereum. We can also distinguish exactly the peers who fully participate in the Ethereum protocol as those who have sent useful block information, i.e. propagate blocks.

We use ethernodes.org to compare the nodes we see and note that there has been work showing how ethernodes.org data is not representative, e.g. many of the peers it shows are not actually running the mainnet Ethereum protocol [37, 60]. They also briefly mention churn, but in no detail. We explore churn in greater detail both in the ethernodes.org data and in our own peer data. Another Ethereum network measurement work is [61] who scrape the P2P layer for peers who they make TCP connections with, though similarly to ethernodes.org, a TCP connection does not distinguish mainnet nodes. They enumerate peer tables for those nodes and analyze their topological properties, though peer tables do not represent actual peer connections on the network.

4.5 Future Work

The basic building block of the P2P layer of blockchain protocols is a mechanism for new peers to bootstrap into the system (i.e. learn about initial peers to connect with), a way to query existing peers about other nodes in the network, a way to store information about those peers and a peer selection protocol for choosing which new connections to make. The peer discovery and selection protocols are vital for insuring an open network where new peers are discoverable to the rest of the network and vise-versa and that peers are selected in such a way so that the network maintains low latency. A side effect of these protocols is also whether they facilitate fair network access for nodes (e.g. certain nodes are not disadvantaged in how quickly they learn about new blocks or their transactions reach miners) and the security of network against different attacks like node eclipsing. Topological information about the network can also facilitate other attacks like network partitioning, censoring and deanonymization.

Our measurement study of the Ethereum network included observations of a large amount of churn in Ethereum peers, a significant fraction of peers not contributing to block propagation and that geolocation plays a significant role in how quickly a client learns about new blocks. In continuation of this study we plan on further exploring how these network characteristics can affect how clients find new peers and the effect this has on their performance and their view of the network.

5 Protocol Design

Two important aspects of the design of blockchain protocols is both their ability to scale and their security which is heavily based on the incentive mechanism for miners to contribute computational power. We approach both questions with new protocol designs below.

5.1 Flyclient: super-light clients for cryptocurrencies

An important aspect of existing cryptocurrencies being able to scale is the ability of users to interact with the system. Unfortunately, most current solutions for verifying cryptocurrency transactions do not suit low-capacity mobile devices such as phones or IoT-devices. Blockchain-based cryptocurrencies use state machine replication that requires nodes to verify every state transition and store the entire state. This is entirely unsuitable for battery, computation, and storage restricted devices. Current mobile solutions often employ a trusted wallet provider which negates much of the benefits of these decentralized ledgers.

To verify that a blockchain is valid without participating in the mining process, a client may choose to download blocks from a miner or a *full node* who holds a copy of the entire chain. Currently, downloading and verifying all blocks in Bitcoin or Ethereum requires a node to download more than 200 GB of data, taking hours to synchronize the node's local blockchain [62]. Such a requirement causes long delays for regular clients and makes it nearly impossible for storage-limited clients to quickly verify transactions.

Light Clients. The original Bitcoin design [1] describes a faster synchronization mechanism, known as *simplified payment verification* that allows lightweight verification of transactions on the blockchain by what is typically referred to as an SPV client (also known as a *light* client [63]).

Instead of downloading all blocks from a full node, an SPV client downloads only the *header* of each block that amounts to a much smaller synchronization overhead than the full blocks (80 bytes versus 1 MB per block in Bitcoin). The block headers are linked together through hashes and include the PoW solutions. This allows an SPV client to verify which chain has the most PoW solutions. Note that light clients neither know whether all transactions are valid nor all consensus rules are being followed. Light clients rather operate under the assumption that the chain with the most PoW solutions follows all rules of the network. This implies that all transactions in this chain are valid and the majority of computation power supports the same valid chain.

Assumption 1 (SPV assumption) *The chain with the most PoW solutions follows the rules of the network and will eventually be accepted by the majority of miners.*

Fortunately, prior work [6, 11, 12, 17] show that this assumption holds as long as an adversary holds only a minority share of the computation power.

Under the SPV assumption, light clients can verify the inclusion of specific transactions in the ledger. This is done by utilizing a Merkle tree commitment of the transactions of a block stored in the block header. A full node provides the light client with an *SPV proof* of the chain along with a Merkle path to the transaction in the tree committed to in the header. Light clients also enable various applications to a broad class of users who need to verify a log of past events recorded on a blockchain, including efficient verification of cross-chain transactions for exchanging cryptocurrencies [64, 65], transferring assets to sidechains [66–68], or sharding blockchains [69, 70] and notary services [71, 72].

Although relying only on block headers reduces the verification overhead of SPV clients, it still incurs a large overhead on resource-limited clients, especially when considering the fact that this overhead increases linearly with the number of blocks. This has already become a major concern in Ethereum due to its significantly-shorter block interval than Bitcoin (∼15 seconds vs ∼10 minutes) and significantly-larger block headers (508 bytes vs 80 bytes). Given that the Ethereum blockchain contains more than 8.2 million blocks (as of July 2019 [73]), an SPV client wishing to verify Ethereum transactions would have to download and store more than 3.9 GB of data. The client has to either download a fresh copy of the data every time it wants to verify a transaction or keep a local copy in its storage and only download the changes since the last synchronization. Either case puts a large burden on the client. The problem is further amplified for users that run clients for multiple blockchains or systems that use sidechains [74].[2]

Sublinear Light Clients. One may wonder if it is possible for a client to verify any event on a blockchain by downloading and/or storing only a sublinear (in the length of the chain) amount of information. In fact, such a performance gain comes with an important security challenge: Since such a client cannot verify every PoW solution in the received blockchain, it can be tricked into accepting an invalid chain by a malicious prover who can precompute a sufficiently-long chain using its limited computational power.

Proposals for sublinear light clients were initially discussed in Bitcoin forums and mailing lists as early as 2012 [76, 77]. Most of them relied on the notion of *superblocks*, blocks that solve a more difficult PoW puzzle than the current target puzzle. Since they appear randomly at a certain rate on an honest chain, the number of superblocks in a chain is a good indicator of the total number of

[2]Ethereum also has a *fastsync* synchronization option which allows a full node to sync to the current chain via SPV [75]. Using this, nodes can start verifying all incoming transactions. Unfortunately, even *fastsync* can take up to 10 hours to receive all headers from the network, likely due to throttling by individual peers.

valid blocks, if miners behave honestly. Kiayias et al. [63] introduced and formalized an interactive proof mechanism, known as *proofs of proof of work (PoPoW)* based on superblocks. PoPoWs allow a prover to convince a verifier with high probability in logarithmic time and communication that a chain contains a sufficient amount of work. In a later work [67], Kiayias et al. provide an attack against the PoPoW protocol and propose a non-interactive and still logarithmic alternative solution known as *non-interactive PoPoW (NIPoPoW)*.

Current Challenges. The superblock-based PoPoW [63] and NIPoPoW [67] suffer from several drawbacks summarized as follows. Both solutions work only if a fixed PoW difficulty is assumed for all blocks. This is not a realistic assumption in practice due the variable combined hashing power of miners in most PoW-based cryptocurrency networks and it is unclear how to modify the superblock based protocols to handle the variable difficulties. Additionally, the variable difficulty setting allows a malicious prover to arbitrarily manipulate block difficulties to perform what is known as a *difficulty raising attack* as described by Bahack [78]. In this attack, the adversary mines fewer but more difficult blocks such that the combined difficulty of the mined blocks exceeds that of honest miners. As a result, the prover can convince the verifier with a fake but seemingly valid chain.

Moreover, the reliance on superblocks makes the protocol susceptible to *bribing* [79] and *selfish mining* [20] attacks. These attacks work by bribing miners to discard superblocks: rational miners accept this if they are paid more than the block reward as superblocks do not yield any extra block reward. The NIPoPoW protocol of [67] defends against this attack but only by reverting to the standard (and expensive) SPV protocol. Finally, NIPoPoW's transaction inclusion proofs are fairly large, even in the optimistic case. This is because such proofs consist of roughly an additional $O(\log(n))$ block headers, where n is the chain length. In some cryptocurrencies such as Ethereum, block headers are quite large, thus resulting in larger NIPoPoW transaction inclusion proofs, e.g., roughly 15 KB in Ethereum.

Our Contribution. In [9] we propose FlyClient, a new blockchain verification protocol for light clients in cryptocurrencies such as Bitcoin and Ethereum. Unlike regular SPV clients that use a linear amount of bandwidth and storage in the length of the chain, FlyClient requires downloading only a logarithmic number of block headers to verify the validity of the chain. Once the chain is verified, the client needs to store only a single block to efficiently verify the inclusion of any transaction on the chain. The FlyClient protocol is a non-interactive PoPoW but overcomes the limitations of the superblock-based NIPoPoW protocol of Kiayias et al. [67].[3] FlyClient is compatible with variable difficulty chains and provides asymptotically and practically-succinct proofs even in the presence of an adversary that can create a $c < 1$ fraction of the honest chain's work. Further, FlyClient requires short transaction-inclusion proofs that consist of only $O(\log(n))$ hashes. In Ethereum, this results in transaction-inclusion proofs that are as small as 1.5 KB which is roughly 10x smaller than NIPoPoWs.

Our protocol is parameterized by $c \in [0, 1)$ and $\lambda \in \mathbb{N}$ such that an adversary which can create forks (of some minimum length) with at most a c fraction of the valid work of the honest chain, succeeds with probability negligible in λ. This corresponds to a slightly stronger and parameterized version of the SPV assumption. The protocol's efficiency depends on both c and λ. We show that FlyClient is efficient even for high values of c (e.g., , $c = 0.9$). Finally, we demonstrate FlyClient's concrete efficiency on Ethereum (see Table 1).

[3]NIPoPoW refers to both a primitive and a protocol (that implements the primitive). Both were introduced by Kiayias et al. [67]. Unless clarified otherwise, we generally use the term NIPoPoW to refer to the superblock-based protocol.

Block Height	10 K	100 K	1,000 K	7,000 K
SPV	4,961	49,609	496,094	3,472,656
FlyClient	161	277	416	524
Improvement	31x	179x	1,275x	6,627x

Table 1: Proof sizes (in KB) for an SPV client and FlyClient in Ethereum at various block heights assuming an adversarial hash power of at most $c = 1/2$ of the honest hash power and failure probability $< 2^{-50}$.

FlyClient achieves these by employing the following techniques:

- **Probabilistic Sampling:** We introduce a PoPoW protocol to randomly sample $O(\log n)$ block headers from a remote blockchain with variable block difficulty. We formally prove the security of our protocol as long as the adversary can only create a c fraction of the honest chain's PoW.
- **Efficient Chain Commitments:** We formalize and use the notion of a *Merkle mountain range (MMR)* [80], an efficiently-updatable commitment mechanism that allows provers to commit to an entire blockchain with a small (constant-size) commitment while offering logarithmic block inclusion proofs with position binding guarantees.
- **Variable Difficulty MMR Checks:** We extend MMRs to contain information about the difficulty and difficulty transitions. This information allows a verifier to efficiently check that the difficulty transitions follow the rules of the chain. Without these checks an adversary could create valid proofs with non-negligible probability by creating few but high difficulty blocks.
- **Non-Interactive and Transferable Proofs:** We introduce a non-interactive variant of FlyClient using the Fiat-Shamir heuristic [81] that allows both the light client and the full node to forward the proof to other light clients without recalculating the proof.

5.2 HaPPY-Mine: designing a mining reward function (On-going work)

Existing cryptocurrencies rely on block rewards for two reasons: to subsidize the cost miners incur securing the blockchain and to mint new coins. Miners in major cryptocurrencies like Bitcoin and Ethereum participate in the protocol by packaging user transactions into blocks and incorporating those blocks into the blockchain. Creating a block involves significant computational power for the *proof of work*, generally iterating over a hash function. This *work*, whether on a CPU, GPU or other specialized hardware, comes at a cost to the miner. To compensate miners for incurring this cost and to incentivize more miners to join, miners collect a *block reward* of newly minted coins for each block that gets added to the blockchain. In expectation, miners are rewarded in proportion to the resources they contribute. This computational work is also what cryptographically ties each block in the blockchain together and makes it so that anyone wanting to *fork* the blockchain, i.e. erase transactions by creating their own version of a subset of the chain, would have to redo an equivalent amount of work. The more resources miners invest in the system, the greater the system hashrate, the more expensive this attack becomes. In effect, the computational work of miners secures the blockchain system by making the blockchain immutable.

There are two common frameworks for the block reward function in terms of distribution of supply. Bitcoin's protocol has a set maximum number of coins that will ever be minted, therefore the mining reward diminishes over time. The mining reward to halves every 210,000 blocks (approximately every 4 years). For now, miners continue to profit since the value of each Bitcoin has increased over time making up for the decrease in reward with increases system hashrate. Eventually though, the mining reward will reach zero and miners will be repaid solely in transaction fees for the transactions they include in the blocks they mine. Ethereum currently has in it's protocol a fixed mining reward of 5 Ethers for all blocks ever. This means that the supply of Ether is uncapped and the mining hashrate can grow linearly in the market value of Ether.

In general miner costs are asymmetric [82] with miners with access to low-cost electricity or mining hardware being at an advantage. This has led to large centralization in both Bitcoin and Ethereum mining, with a significant portion of the hashrate being controlled by a few mining pools [83–85]. This prevents other players from having a share of the market. We ask the question, can we design a mining reward function that alleviates these problems?

Our Contribution. We develop a novel hashrate-based mining reward function,HaPPY-Mine, which sets the block reward based on the system hashrate. HaPPY-Mine is defined so that as the system hashrate increases, the block reward smoothly decreases. In this work so far we have done the following:

- We introduce the notion of a *hashrate-pegged mining reward function*, and formally argue that it can help in decentralizing the blockchain by reducing the hashrate that a new miner is incentivized to buy.
- We present HaPPY-Mine, a family of hashrate-pegged mining reward functions that dispense rewards in proportion to the expended hashrate. We conduct a rigorous equilibrium analysis of the HaPPY-Mine family under general miner costs. We establish that equilibria always exist, and are more decentralized than an equilibrium under the static reward function: in particular, HaPPY-Mine equilibria have at least as many participating miners and lower total hashrate than an equilibrium for the static reward function.
- We show that HaPPY-Mine equilibria (as well as that of a static reward function) are resistant to any collusion attack involving fewer than half the miners, and that a Sybil attack does not increase the utility of the attacker.
- We consider the scenario where rewards are issued in the currency of the blockchain and study the effect of a change in the currency's value on the equilibrium. We show that in HaPPY-Mine, an increase in the value of the cryptocurrency allows more higher cost miners to participate, resulting in greater decentralization than the equilibrium under the static reward function.

5.2.1 Related Work

As stated above, HaPPY-Mine is an example of the generalized proportional model of [86]. We compare HaPPY-Mine with the equilibrium of the static reward function of [87] associated with most cryptocurrencies. Other papers have looked at different games involved in mining including the game between participants in mining pools and different reward functions for how the pool rewards are

allocated [88]. In [89], the authors present a continuous mean-field game for bitcoin mining which captures how miner wealth and strategies evolve over time. They are able to capture the "rich get richer" effect of initial wealth disparities leading to greater reward imbalances. [90] models the blockchain protocol as a game between users generating transactions and miners. They show if there is no block reward, then there is an equilibria of transaction fee and miner hashrate. Higher fees incentivize higher miner hashrate which leads to smaller block times. When you introduce a high static block reward, the users may no longer be incentivized to introduce mining fees and there may no longer be an equilibrium.

In contrast, [91] also studies the case where there is no block reward, and analyzes new games in which miners may use transactions left in the mempool (pending transactions) to incentivize other miners to join their fork. Another work exploring the mining game when there is no block reward is that of [92] who introduce *the gap game* to study how miners choose periods of times when not to mine (gaps) as they await more transactions (and their fees). They show that gap strategies are not homogeneous for same cost miners and that the game incentivizes miner coalitions reducing the decentralization of the system.

Previous work on rational attacks in cryptocurrency mining includes [93] who study the security of Bitcoin mining under rational adversaries using the Rational Protocol Design framework of [94] as a rational-cryptographic game. Also, [95] who analyze the Bitcoin mining game as a sequential game with imperfect information, and [96] analyze selfish mining by looking at the minimal fraction of resources required for a profitable attack. In [97], the authors explore the game of Bitcoin mining cost and reward focusing on incentives to participate honestly. They outline the choices different players can make in a blockchain system and their possible consequences. Another work related to the incentives at play in cryptocurrency mining is [98] which looks at the coordination game of Bitcoin miners in choosing which fork to build on when mining. They find the longest chain rule is a Markov Perfect equilibrium strategy in a synchronous network and explore other miner strategies, some that result in persistent forks.

5.3 Future Work

In this work so far we have analyzed our HaPPY-Mine protocol by looking at the equilibrium strategy for miners as a one shot game. We've been able to prove the equilibria are collusion and sybil proof and have properties which make it more decentralized than the standard static reward function. A natural extension of this is to consider properties of HaPPY-Mine over time, including at the inception of the coin and as it gains value. Extending our analysis to an evolving game would allow us to make broader security guarantees as well as to incorporate market share control into our analysis.

6 Research Plan

Table 2 presents my plan for completion of the research outlined in this proposal. The plan includes wrapping up the research of Section 4.3 and Section 5.2 to be presented in Financial Cryptography in Spring 2021. As presented in Section 4.5, our proposed next project is to look deeper into the impact of network churn on the performance of blockchain protocols. Time permitting, we also plan to extend our Markov-based security analysis of blockchain protocols as presented in Section 3.3. The proposed plan is to wrap-up these projects by the end of Fall 2021 and move on to thesis writing

to defend by the end of the Spring 2022.

Timeline	Work	Progress
HotNets'17	Ethereum Fork Measurement	Published
IMC'18	Ethereum Contract Topology Measurement	Published
CCS' 18	Markov Model for Blockchain Protocols	Published
S&P'20	Flyclient: A light-client protocol	Published
FC'21 (accepted)	Ethereum Network Measurement	Ongoing
FC'21 (accepted)	HaPPY-Mine Mining Reward Function	Ongoing
Spring-Fall 2021	Effects of Churn on Blockchain Networks	Ongoing
Spring 2022	Thesis writing	-
May 2022	Thesis defense	-

Table 2: Plan for completion of my research

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] "Ethereum market capitalization," https://coinmarketcap.com/currencies/ethereum/, 2020.

[3] https://ens.domains.

[4] https://medium.com/mit-media-lab-digital-currency-initiative/
medrec-electronic-medical-records-on-the-blockchain-c2d7e1bc7d09.

[5] https://chromaway.com/papers/A-blockchain-based-property-registry.pdf.

[6] L. Kiffer, R. Rajaraman, and A. Shelat, "A better method to analyze blockchain consistency," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 729–744.

[7] L. Kiffer, D. Levin, and A. Mislove, "Stick a fork in it: Analyzing the ethereum network partition," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 94–100.

[8] ——, "Analyzing ethereum's contract topology," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 494–499.

[9] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 928–946.

[10] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology — CRYPTO' 92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings*. Springer Berlin Heidelberg, 1993, pp. 139–147. [Online]. Available: http://dx.doi.org/10.1007/3-540-48071-4_10

[11] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications." in *EUROCRYPT (2)*, 2018, pp. 281–310.

[12] 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*. Springer, 2017, pp. 643–673.

[13] https://www.bitcoincash.org/.

[14] "Bitcoingold," https://bitcoingold.org.

[15] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[16] "Ethereum wire protocol (eth)," https://github.com/ethereum/devp2p/bl
ob/master/caps/eth.md.

[17] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, 2017, pp. 291–323.

[18] Q. Martino and Popejoy, "Chainweb: A proof-of-work parallel-chain architecture for massive throughput," May 2018.

[19] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[20] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[21] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.

[22] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[23] Y. Sompolinsky and A. Zohar, "PHANTOM: A scalable blockdag protocol," Cryptology ePrint Archive, Report 2018/104, 2018, https://eprint.iacr.org/2018/104.

[24] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1019, 2015.

[25] ——, "On trees, chains and fast transactions in the blockchain." *IACR Cryptology ePrint Archive*, vol. 2016, p. 545, 2016.

[26] R. Pass and E. Shi, "The sleepy model of consensus," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 380–409.

[27] J. Chen and S. Micali, "Algorand," https://arxiv.org/abs/1607.01341, 2016.

[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *SOSP'17*, 2017.

[29] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," *arXiv preprint arXiv:1605.07524*, 2016.

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

[31] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[32] https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki.

[33] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," *et al.*, 2015.

[34] A. Hertig, "$160 million stuck: Can parity still shake up ethereum?" https://www.coindesk.com/startup-lost-160-million-still-wants-shake-ethereum/.

[35] D. Siegal, "Understanding the dao attack," https://www.coindesk.com/understanding-dao-hack-journalists/.

[36] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," *arXiv preprint arXiv:1801.03998*, 2018.

[37] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring ethereum network peers," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 91–104.

[38] L. Anderson, R. Holz, A. Ponomarev, P. Rimba, and I. Weber, "New kids on the block: an analysis of modern blockchains," *arXiv preprint arXiv:1606.06530*, 2016.

[39] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: identification, analysis, and impact," *Future Generation Computer Systems*, vol. 102, pp. 259–277, 2020.

[40] F. Victor and B. K. Lüders, "Measuring ethereum-based erc20 token networks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 113–129.

[41] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[42] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Internet Measurement Conference*. ACM, 2013, pp. 127–140.

[43] S. Ranshous, C. A. Joslyn, S. Kreyling, K. Nowak, N. F. Samatova, C. L. West, and S. Winters, "Exchange pattern mining in the bitcoin transaction directed hypergraph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 248–263.

[44] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the bitcoin utxo set," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 78–91.

[45] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 34–51.

[46] T. Neudecker and H. Hartenstein, "Could network information facilitate address clustering in bitcoin?" in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 155–169.

[47] M. Moser, "Anonymity of bitcoin transactions: An analysis of mixing services," in *Münster Bitcoin Conference (MBC)*, 2013.

[48] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *International conference on financial cryptography and data security*. Springer, 2015, pp. 44–61.

[49] ——, "Analyzing the bitcoin ponzi scheme ecosystem," in *Bitcoin Workshop*, 2018.

[50] H. Basil Al Jawaheri, M. Al Sabah, and Y. Boshmaf, "Measurement and analysis of bitcoin transactions of ransomware," in *Qatar Foundation Annual Research Conference Proceedings*, vol. 2018, no. 3. HBKU Press Qatar, 2018, p. ICTPD1026.

[51] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). Springer*, 2018.

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

[53] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 469–485.

[54] M. Yabandeh, N. Knezevic, D. Kostic, and V. Kuncak, "Crystalball: Predicting and preventing inconsistencies in deployed distributed systems." in *Symposium on Networked Systems Design and Implementation*. USENIX, 2009, pp. 229–244.

[55] P. Agrawal, "Fault tolerance in multiprocessor systems without dedicated redundancy," *IEEE Transactions on Computers*, vol. 37, no. 3, pp. 358–362, 1988.

[56] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.

[57] R. Norvill, B. B. F. Pontiveros, R. State, I. Awan, and A. Cullen, "Automated labeling of unknown contracts in ethereum," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017, pp. 1–6.

[58] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: platforms, applications, and design patterns," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 494–509.

[59] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[60] "Measuring ethereum nodes," https://medium.com/coinmonks/measuring-ethereum-nodes-530bfff08e9c.

[61] Y. Gao, J. Shi, X. Wang, Q. Tan, C. Zhao, and Z. Yin, "Topology measurement and analysis on ethereum p2p network," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7.

[62] "Blockchain takes way too long to sync · issue #2394 · ethereum/mist," https://github.com/ethereum/mist/issues/2394, 2017, (Accessed on 11/29/2018).

[63] A. Kiayias, N. Lamprou, and A.-P. Stouka, *Proofs of Proofs of Work with Sublinear Complexity*. Springer Berlin Heidelberg, 2016, pp. 61–78.

[64] M. Herlihy, "Atomic cross-chain swaps," *arXiv preprint arXiv:1801.09515*, 2018.

[65] "ethereum/btcrelay: Ethereum contract for bitcoin spv," https://github.com/ethereum/btcrelay, 2018, (Accessed on 12/14/2018).

[66] A. Back and G. Maxwell, "Transferring ledger assets between blockchains via pegged sidechains," Nov 2016, uS Patent App. 15/150,032.

[67] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 505–522.

[68] A. Kiayias and D. Zindros, "Proof-of-work sidechains," Cryptology ePrint Archive, Report 2018/1048, 2018, https://eprint.iacr.org/2018/1048.

[69] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *2018 ACM Conference on Computer and Communications Security (CCS)*, 2018.

[70] 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 (S&P)*, 2018, pp. 19–34. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2018.000-5

[71] "Open timestamps," https://opentimestamps.org/, 2018.

[72] "Stampery," https://stampery.com/, 2018.

[73] "Ethereum blocks," https://etherscan.io/blocks, July 2019, (Accessed on 07/30/2019).

[74] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, 2014.

[75] "Getting deep into geth: Why syncing ethereum node is slow," https://bit.ly/2OOMcXC, July 2018.

[76] socrates1024, "The high-value-hash highway," https://bitcointalk.org/index.php?topic=98986.0, 2012.

[77] M. Friedenbach, "Compact spv proofs via block header commitments," https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg04318.html, March 2014.

[78] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013.

[79] J. Bonneau, "Why buy when you can rent? bribery attacks on bitcoin-style consensus," in *Proceedings of Financial Cryptography*, 2016.

[80] P. Todd, "Merkle mountain range," https://github.com/opentimestamps/opentimestamps-server/blob/master/doc/merkle-mountain-range.md, 2012.

[81] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 186–194.

[82] "Here's how much it costs to mine a single bitcoin in your country," https://www.marketwatch.com/story/heres-how-much-it-costs-to-mine-a-single-bitcoin-in-your-country-2018-03-06.

[83] "Pool distribution," https://btc.com/stats/pool?pool$_m ode = month3$.

[84] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, ""is bitcoin a decentralized currency?"," *IEEE security & privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[85] "Top 25 miners by blocks," https://etherscan.io/stat/miner?blocktype=blocks.

[86] X. Chen, C. Papadimitriou, and T. Roughgarden, "An axiomatic approach to block rewards," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 124–131.

[87] N. Arnosti and S. M. Weinberg, "Bitcoin: A natural oligopoly," in *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[88] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden, "Incentive compatibility of bitcoin mining pool reward functions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 477–498.

[89] Z. Li, A. M. Reppen, and R. Sircar, "A mean field games model for cryptocurrency mining," *arXiv preprint arXiv:1912.01952*, 2019.

[90] E. Iyidogan, "An equilibrium model of blockchain-based cryptocurrencies," *Available at SSRN 3152803*, 2019.

[91] 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*, 2016, pp. 154–167.

[92] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, 2018, pp. 713–728.

[93] C. Badertscher, J. Garay, U. Maurer, D. Tschudi, and V. Zikas, "But why does it work? a rational protocol design treatment of bitcoin," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2018, pp. 34–65.

[94] J. Garay, J. Katz, U. Maurer, B. Tackmann, and V. Zikas, "Rational protocol design: Cryptography against incentive-driven adversaries," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 648–657.

[95] J. Beccuti, C. Jaag *et al.*, "The bitcoin mining game: On the optimality of honesty in proof-of-work consensus mechanism," *Swiss Economics Working Paper 0060*, 2017.

[96] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.

[97] 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*, vol. 2013, 2013, p. 11.

[98] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta, "The blockchain folk theorem," *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 2019.