Can open decentralized ledgers be economically secure?

(Preliminary draft)

Jacob D. Leshno* Rafael Pass† Elaine Shi‡

Abstract

Traditional payment processors are the subject of antitrust concerns and regulations. Open decentralized ledgers (e.g., Bitcoin) provide an alternative. They do not rely on a central authority, avoiding antitrust and monopoly concerns. However, the open nature of these systems gives rise to many challenges, including fundamental questions about their security.

To address this question, we consider a framework that combines economic theory and distributed systems theory and define *economic security* for general permissionless decentralized ledgers. Analysis of Bitcoin's Nakamoto protocol shows that block rewards are ineffective in providing economic security due to limitations of incentives in environments with many anonymous participants. We present an alternative protocol showing that an open decentralized ledger can be economically secure.

1 Introduction

Payment processing is necessary in a modern economy. It is also hugely profitable for payment processing firms who collectively collected global revenues of $2.1 trillion in 2021 [McKinsey and Company, 2021]. These revenues are enabled by limited competition due to network effects and barriers to entry, giving many payment-processing firms significant market power. Payment providers are often subject to regulations and antitrust litigation that attempts to mitigate their market power.

Open distributed ledgers provide an alternative. Blockchains like Bitcoin and Ethereum provide payment processing to users without relying on any particular party that controls the system. These blockchains are operated by computer nodes (miners) that are interchangeable and can freely enter and exit. This design removes pricing power from miners and protects users from the harm of monopoly pricing.

However, the open design creates security challenges. If the system allows anyone to become a node that participates in the operation of the protocol, so can a malicious attacker. Such

*University of Chicago, Booth, Jacob.Leshno@ChicagoBooth.edu. This work is supported by the Robert H. Topel Faculty Research Fund at the University of Chicago Booth School of Business.

†Cornell Tech, Computer Science, rafael@cs.cornell.edu.

‡Carnegie Mellon University, Computer Science and Electrical and Computer Engineering, runting@gmail.com.

concerns are exemplified by the vulnerability of Bitcoin's Nakamoto consensus to corrupt majority attacks (also called 51% attacks, or double spend attacks), which gained a lot of attention in the economic literature including including [Budish, 2022], [Auer, 2019], [Bonneau, 2016b], [Gans and Halaburda, 2023], [Chiu and Koeppl, 2022], [Moroz et al., 2020], [Garratt and van Oordt, 2020], and [Gans and Halaburda, 2023]. Although Bitcoin itself was not attacked, attacks occured against other similar systems [Nesbitt, , Kelso, , zen,].

Motivated by these works, we ask whether it is possible to create an open distributed ledger that is secure.

We first define economic security by combining the canonical framework from the distributed systems literature with an economic framework. Consider a merchant who wishes to receive payment. In traditional payment systems, the merchant relies on security measures to detect invalid payments. In a distributed ledger, the merchant relies on a payment terminal that communicates with the network to verify whether the payment is valid. In distributed systems terminology, the merchant problem is achieving consensus on the validity of the payment. If the distributed system satisfies consistency, the merchant cannot be deceived. We therefore define the economic security of a distributed ledger as the economic cost required to violate consistency. (See Section 2 for the precise definition and background on distributed system).

We then apply our definition to Bitcoin's Nakamoto consensus protocol and evaluate the use of miner rewards to secure the ledger in a model with profit motivated miners. Two deficiencies of incentives render the system insecure. First, the protocol fails to distinguish between an attacker and honest miners, allowing attackers to avoid punishment. Second, even if following the attack the community takes actions that harm the miners (e.g., exchange rate collapse), such actions harm all miners whether or not they participated in the attack. If miners are small, the tragedy of the commons can arise.

Given there were no attacks on Bitcoin, we consider in Section 5 other factors that deter attacks. One important factor is that the community will be able to detect the attack.[1] The community can completely overriding the protocol by a "community hard fork" which manually changes the ledger to its correct pre-attack state. Although this response requires significant coordination and nontrivial effort by the community, it undoes any effect of the attack and provide deterrence against attacks.

Somewhat surprisingly, we find that it is possible for an open distributed ledger to be economically secure, and we present such a protocol. The intuition for the protocol is that the Nakamoto protocol could be made secure if we could emulate the community hard fork within the protocol. Our protocol, which we call Stubborn Nakamoto, is a modification of the Nakamoto protocol. The protocol requires essentially the same assumption as the Nakamoto protocol and operates in the same manner when the Nakamoto protocol operates correctly. It differs from the Nakamoto protocol in that it guarantees consistency under corrupt majority, which establishes its economic security.

[1] It is commonly assumed that an attack will trigger a collapse of bitcoin's price, implicitly assuming that markets will be aware of the attack.

Essentially, our Stubborn Nakamoto attempts to detect consistency violations of the Nakamoto protocol and either avoids them or transforms them into liveness violations.

The remainder of the paper is organized as follows. Section 2 motivates and provides our definition of economic security and overviews required concepts from distributed systems. Section 4 presents our definitions and provides the relevant background on Bitcoin's Nakamoto consensus protocol. Section 4 analyzes the economic security of Nakamoto and shows the deficiencies of incentives. Section 5 discusses other factors that deter attacks in practice. Section 6 presents the Stubborn Nakamoto Protocol and proves its security. Technical details, formal definitions and proofs are in Appendix A.

1.1 Additional Related Work

The idea of checkpointing in consensus has also been used in various earlier works [Daian et al., 2019, Buterin and Zamfir, 2015, Karakostas and Kiayias, 2021, Neu et al., 2021, wea,]. However, our usage of checkpointing is different from previous works, in that we formally guarantee consistency under any number of Byzantine corruptions, and guarantee liveness under honest majority.

While we focus on analyzing the economic security of Bitcoin's Nakamoto consensus protocol, it is worth noting that some proof-of-stake consensus protocols can provide Byzantine forensics and accountability [Sheng et al., 2021, Neu et al., 2022]. In other words, if nodes misbehave, it may be possible to produce cryptographic evidence that implicate them. Such cryptographic forensics evidence can serve as a deterrent to attacks for these proof-of-stake protocols.

2 Framework for Economic Security of Ledgers

2.1 Background: digital ledgers, cryptography and distributed computation

Before we give our definition, we first outline the challenges that an electronic ledger must solve, and how cryptography and distributed protocols address these challenges. Doing so will motivate our framework and definition of economic security.

As an illustration, consider that our goal is to implement a digital payment system. That is, we want a system that records account balances, allows users to observe their balance, and allows users to transfer funds from one account to another. The system should follow standard accounting rules: Balances must be non-negative. A transfer credits one account by the amount debited from another account. A transfer of funds must be authorized by the debited account holder, and the debited account must have sufficient funds for the transfer.

With a trusted record keeper it is straightforward to implement a payment system. The record keeper authenticates that each transfer is authorized by the account holder, and updates the balances after each such transaction. The record keeper can record balances using a standard ledger: an excel spreadsheet, a more sophisticated computer database, or even ancient bookkeeping methods. The trusted record keeper is the definitive authority that certifies the current balance held in each account.

Without the trusted record keeper, two challenges arise. First, all must agree on account balances, and maintain agreement as these accounts are updated. How can we generate agreement on the ledger without a definitive authority? Second, the ledger must ensure that standard accounting rules are enforced, and that any withdraw from an account is authorized by the accounts owner. How can we ensure that the ledger accurately records balances? We detail below how the second challenge can be solved using a publicly available ledger and cryptographic signatures, and how distributed computing protocols address the first challenge.

2.1.1 Background: Cryptographic Signatures

As a stepping stone, assume that we have a *public write-only ledger* which allows anyone to read or append their data at the end of the ledger.[2] For example, imagine a village with a big stone at its center that anyone can carve their data to.

Using a public write-only ledger together with cryptographic signatures we can construct a payment system as follows. Associate each account with a public key. Ownership of an account is defined to be knowledge of the corresponding private key, which can be proved by cryptographic signatures.[3] Initialize accounts to specified balances.

If a user wishes to make a transfer between accounts, the user writes a transaction to the ledger that specifies debited accounts and the corresponding amounts to debit, credited amounts and the amounts to credit, and a cryptographic signature showing ownership of the debited accounts. All transactions are written to the publicly accessible ledger, allowing anyone to read the ledger, calculate current balances for all accounts, and verify that standard account rules are followed. In addition, anyone can verify that a transaction is authorized the debited account holder, and that the accounts holds a sufficient balance for the transfer. If a transaction violates accounting rules or is not properly signed, that transaction is ignored by all readers. The resulting ledger is a payment system that maintains and updates account balances without relying on a trusted record keeper.

2.1.2 Background: Distributed Consensus

Distributed consensus provides a way to implement a public write-only ledger without relying a single trust record keeper. Loosely speaking, a distributed protocol allows a collection of computers to behave as if they were one system. Such distributed systems can achieve greater scale and higher resiliency to computer failures, while appearing to users as if they are one reliable computer. For example, a firm may run its website on multiple replicated servers; users can connect to any server and see the same website and data; and the website remains accessible even if one of the servers crashes.

[2] That is, this public write-only ledger solves the first problem of maintaining an agreed-upon ledger.

[3] The public key can be used to verify that a signature on a message is a valid cryptographic signature. An agent who knows of the private key can generate a signature for any message through a simple computation. It is a standard cryptographic assumption that an agent who does not know the private key cannot create a valid signature (given plausible limitations on computational resources).

The canonical problem in distributed systems is state machine replication, in which a set of nodes collectively maintain a ledger. Each node can only observe its local state, it learns about others only through the messages it receives. Nodes repeatedly attempt to achieve consensus on a value to be recorded in the ledger. Initially, a node may be uncertain as to the value collectively record. A node may send messages to others or wait for messages before deciding. A node is said to *finalize* a record once it is certain of the record's value. Once the node finalized a record in the ledger, that record cannot be undone or changed. For example, different web servers may communicate with each other to determine a shipping record. A server must finalize the shipping record before sending it to a user to ensure the user never sees a wrong or outdated shipping record.

A distributed protocol is a strategy to each node, specifying the action the node should take given any potential local state. Nodes are said to be *honest* if they follow the protocol, and are said to be *faulty* or *corrupt* otherwise. Proper behavior of the protocol is captured by the following properties.[4] We give an informal description here, see appendix A for formal definitions.

- **Consistency** - if an honest node finalized a record, any honest node agrees with that finalized record.
- **Liveness** - new records become finalized (within a reasonable amount of time).

When both of these properties hold, the collection of nodes acts as one record keeper that maintains a "consensus ledger". These properties are also useful in distinguishing different possible failures. If the system stops processing records, it is a liveness violation. An example is a payment systems that stops processing transactions because it lost communication. If two nodes disagree on a finalized record, it is a consistency violation. An example is a payment app giving the user confirmation a payment was processed although the payment was not processed.

2.2 Definition for economic security of a ledger

To motivate our definition, consider a merchant who wishes to provide a good to a buyer in exchange for payment. A dishonest buyer may want to deceive the merchant into thinking that payment was provided, while not actually paying the merchant. For example, a merchant might be reluctant to accept paper bills that might be counterfeit, or paper checks that may not be backed by sufficient funds. The merchant may employ some security measures, for example checking security devices on the bills or calling the bank to verify a check will be honored. A traditional payment is secure if merchants can protect themselves from fraud by following appropriate security measures.

Likewise, a merchant using a digital payment system can employ security measures. The merchant will use some computer terminal to verify the payment. In a centralized payment system operated by a trusted record keeper (e.g., a bank), the merchant's terminal can communicate with the trusted record keeper to verify the payment. In an open decentralized ledger there is no single

[4]These properties are often accompanied with a *validity* property that requires the record to be a valid value (for example, a recorded value must be a value submitted by some user). This requirement can be incorporated into the liveness property.

trusted record keeper that can verify the payment. If the payment system operates on an open decentralized ledger, the merchant's terminal will have to communicate with the collection of nodes operating the ledger to verify the payment is included in the consensus ledger.

Using terminology from distributed systems theory, the merchant is attempting to reach consensus with the payment system on the validity of the payment. The merchant's payment terminal is a consensus node.[5] The merchant should consider the payment valid if its node finalized the transaction and the transaction is valid given the node's local record. Assuming the merchant's node follows the protocol (i.e., is honest node), a deceit of the merchant is a violation of the distributed system's consistency. Vice versa, the the distributed system's consistency guarantees that if the merchant's honest node finalized a transaction, the transaction is considered valid by the remainder of the system.

Classic protocols in distributed systems guarantee consistency and liveness when a sufficient majority of the nodes are honest. Such classic protocols were designed for permissioned settings, for example a firm with multiple data centers, in which faulty behavior of multiple nodes is a rare event. But in open decentralized protocols nodes are operated by profit motivated agents that can enter and exit at will. An attacker can start many nodes that participate in the protocol, or create financial incentives to convince existing nodes to become faulty. In other words, whether the majority of nodes are honest or corrupt is determined endogenously.

We therefore define the economic security of an distributed ledger to be the minimal attacker's cost required to violate consistency. This definition gives a meaningful assurance to users of the system that it is difficult for an attacker to deceive agents.

To measure the economic security of a distributed ledger, we need to specify the assumptions on the distributed systems model (e.g., communication delays), as well as the economic environment that stipulates potential actions of the attacker and their associated costs.[6]

Definition 1. *Given an distributed systems model and economic environment, the economic security of a distributed ledger is the minimal attacker's cost required to violate consistency.*

Note our definition takes a similar approach to [Budish, 2022] in that we focus on the attacker's cost and we do not attempt to quantify the attacker's gain. the attacker's costs can be quantified within the model, whereas the benefits to an attacker may depend on many aspects that cannot be observed within the model.[7] By focusing on the attacker's costs we can provide users of the system with a guarantee that does not hinge on the motives of the attacker.

Given reasonable assumptions on the attacker's benefit, our definition informs a merchant using the system as to how to adjust their behavior to ensure that an attacker cannot profitably deceive them. For example, a merchant may limit payments to a value below the cost of an attack.

[5]The merchant may employ an observer node that maintains a local record of the ledger and runs the protocol to reach consensus, but does not attempt to add new data to the ledger. For example, an observer node in Bitcoin's Nakamoto consensus acts like a miner except that it does not attempt to mine new blocks.

[6]If we want to show that a protocol has infinite economic security, we just have to show that no possible strategy can break consistency (and in this case we may not need to specify the precise costs of each action).

[7]For example, the attacker may have financial interests stemming from bets made outside the system.

2.3 Examples of profitable consistency violations

Consistency violations lead to misinformed parties, allowing attackers to profit from defrauding counterparties. We give a few examples.

Double spend: The commonly considered double spend attack is a form of consistency violation. In it, the attacker sends the same funds to two different parties in two different transactions, but only one of the two can be valid. If consistency holds, the receiving parties will be able to detect which transaction is valid. Vice versa, if the attacker is able to deceive a merchant into accepting an transaction that conflicts with the consensus ledger, then the attacker violated consistency.

History rewrite: Consistency requires that any nodes present record agrees with any data finalized in the past. In particular, it implies that ledger cannot revert finalized transactions. A dishonest attacker can significantly gain if they are able to undo or rewrite finalized transaction. For example, consider an attacker selling an option on the oil price. If the option is a form of commitment, enforced by a transaction recorded and finalized in the ledger. At the option's maturity date the attacker will learn whether selling the option turned out up profitable, and will want to undo the sale it was not profitable. Retroactively removing the recorded transaction from the ledger is a consistency violation.

3 Background: Bitcoin's Nakamoto Protocol

3.1 Background - The Nakamoto protocol

This subsection introduces our notation and provides a short summary of the Nakamoto protocol [Nakamoto, 2009] used by Bitcoin (and many other protocols). We focus on design elements relevant for attacks (in particular, the protocol's behavior under network partition).

The protocol aims to implement a write-only ledger maintained by a network of anonymous parties. We refer to these parties as *miners* or *nodes* interchangeably. Each node communicates with others, and maintains a local copy of the ledger.

The Blockchain Data Format The ledger is saved as a *blockchain* that grows by appending new blocks over time. A blockchain is an ordered sequence of blocks $C(b_n) = (b_0, b_1, \ldots, b_n)$, starting with the genesis block b_0. Each block b_ℓ contains transaction data, as well as a cryptographic pointer to the preceding block $b_{\ell-1}$. The final block in the chain b_n uniquely identifies the entire chain $C(b_n)$. For each block b_ℓ we use $h(b_\ell) = \ell$ to denote the height of the block, defined as the number of the blocks in the chain $C(b_\ell) = (b_0, b_1, \ldots, b_\ell)$ minus one.

Each node i maintains a local record of the blocks it received $R_i = \{b_\ell\}$. This record does not necessarily corresponds to a single blockchain, as it may contain multiple blocks of the same height (i.e., blocks that cannot be in the same chain). Given some chain $C(b_n) = (b_0, b_1, \ldots, b_n)$, we say that b_{n-k} has confirmation depth $k+1$, or the block b_{n-k} is k-deep in the chain $C(b_n)$.

Henceforth let D denote some mining difficulty parameter, which also means the expected number of hashes needed to successfully mine a block.

The Nakamoto Protocol The protocol uses a hash function to randomly select a node to add a block to the blockchain. Each node composes a suggested block of transaction data, selects a nonce value[8] and computes the block's hash which is a number between $[0, 2^\ell - 1]$ for some integer ℓ also referred to as the length of the hash. If the value of the hash is above the difficulty threshold, the miner can select a different nonce and try again. If the value of the hash is below the difficulty threshold $2^\ell/D$, the block is legal. The node is said to have mined a block, and the new block is communicated to all other nodes. A node's mining power is the number of hash computations the node can attempt in a round.

A node that mines a block transmits it to all other nodes. Once a node i receives a block, it verifies the block is legal, adds the block to its local record R_i and communicates it to all other nodes. The protocol asks nodes to mine blocks that extend the *longest chain* in their local record.[9] Ties are broken arbitrarily.

To compensate nodes for their costs, a node who mines a block can include a special transaction in its block that awards the node with a fixed number of newly minted coins as well as transaction fees from transaction included in its block. We collectively refer to all of these as the block reward and use p_b to denote its USD value.

To finalize blocks in the Nakamoto protocol, we need some cutoff parameter henceforth denoted k. Any block that is at least k-deep in the current longest chain is considered finalized. In Bitcoin, the value $k = 6$ is commonly used.

Following the literature [Garay et al., 2015, Pass et al., 2017, Shi,], , we model the hash function as a random oracle that always returns a random integer between $[0, 2^\ell - 1]$ on a fresh input. We may assume that the execution proceeds in (possibly infinitesimally small) rounds, and in each round, each unit of mining power can invoke the hash function once. In practice, the difficulty threshold is periodically adjusted so that the total block arrival rate is kept roughly constant. We defer more details on the formal execution model to Appendix A.

3.2 Consistency and Liveness of the Nakamoto protocol in distributed systems

To illustrate how the Nakamoto protocol works and the challenges it faces, consider first an idealized version of the protocol in which all nodes are honest and messages are communicated instantly. Under these conditions exactly one block is mined block at each height. The local record of any node is an identical chain of blocks, and nodes achieve consensus on this chain.

But if messages are communicated with some lag, it is possible for two honest nodes to "simultaneously" mine conflicting blocks of the same height. For example, assume that it takes 10

[8]The nonce value is a field that serves the sole purpose of allowing multiple inputs to the hash function for the same block.

[9]That is, if $b^* \in \arg\max\{h(b) \mid b \in R\}$ then $C(R) = C(b^*)$ is the longest chain given the node's local record R, and the node should attempt to mine a block that points to b^*.

seconds for a message from node i to reach node j. Suppose that at time t all miners attempt to extend block b_{n-1}, and that node i mines a block b_n at time t and communicates it immediately to all other nodes. At time $t+5$ honest node j is yet unaware of the block b_n mined by i, and can mine block b'_n (that conflicts with block b_n). At time $t+20$ the local record of any node includes two conflicting blocks b_n, b'_n, and it is unclear which block should be considered part of the consensus chain.

The Nakamoto protocol addresses this challenges using the *longest chain rule* to reach consensus among nodes on a unique chain. At each point in time, honest nodes attempt to extend the longest chain in their record. If there are two longest chains, honest nodes try to extend one of them. If network lags are sufficiently short (relative to inter-block time), then the probability that two honest miners mine conflicting blocks "simultaneously" is small, and the probability that two conflicting chains $(b_0, \ldots, b_{n-1}, b_n, \ldots, b_{n+k})$ and $(b_0, \ldots, b_{n-1}, b'_n, \ldots, b'_{n+k})$ are mined by honest nodes is exponentially small in k. Once a block b is part of the longest chain and is k confirmed, it is extremely likely that the block b is a part of the longest chain of any honest miner from that point forward.

The formal proof (which can be found in [Garay et al., 2015, Pass et al., 2017, Shi, , Guo and Ren, 2022]) shows that Nakamoto consensus satisfies consistency and liveness if (i) network lags are bounded and sufficiently short, and (ii) a sufficient fraction of the mining power is controlled by honest nodes — see Appendix A for a more precise statement of the "sufficient" condition. The proof formalizes the above intuition under these conditions, showing even if a malicious attacker controls a fraction of the mining power, blocks that are k confirmed are extremely likely to be part of any node's longest chain.

Both assumptions are necessary: the Nakamoto protocol can lose consistency if network lags are too long, or if the adversary controls enough computational power. To see why, consider a network partition. Suppose that miners are either in North America (NA) or South America (SA), and that a network partition prevents any communication between NA and SA for a period of time. The protocol directs each group of miners to continue mining and extend the longest chain in their local record. Given the partition, two separate chains grow, one by NA nodes and one by SA nodes. When the network partition is resolved, both NA nodes and SA nodes become aware of both chains. At which point the protocol reconciles the two conflicting chains by asking all nodes to adopt the longest chain. If the partition lasts long enough, this reconciliation can violate consistency: NA and SA will mine conflicting blocks that are k confirmed, and the group with shorter chain will later discard blocks that are k confirmed.

An attacker with sufficient computational power can exploit the longest chain rule by "pretending" to be partitioned. If the attacker is able to mine a conflicting chain that is longer than the honest chain, the protocol directs all miners to adopt the new longest chain. Although one may be suspicious that a conflicting chain exists (as honest nodes should not be mining a conflicting chain), the protocol cannot detect whether the conflicting chain was mined by a malicious actor, or whether it was mined by an honest miner that was partitioned from the network.

In practice, the community will be aware the protocol was attacked, as it is very unlikely that a significant fraction of the compute power in the network is partitioned.[10] A common assumption in the literature that an attack on the protocol will trigger a collapse of the exchange rate of the protocol's coin, which implicitly assumes that the community is able to detect attacks.

4 Economic Security of Nakamoto Consensus

Bitcoin's Nakamoto protocol showed the possibility of permissionless consensus: a ledger operated by anonymous profit driven miners that participate at will. These properties enable free entry and exit of miners and protect its users from monopoly's harm [Huberman et al., 2021]. However, economic theory suggests that these "decentralization" properties can make it difficult to provide miner with incentives to deter attacks on the system.

In this section we apply our definition to Bitcoin's Nakamoto protocol in two economic environments to show two deficiencies of the protocol's security incentives. First, the protocol fails to punish attackers because the protocol does not distinguish between attackers and honest miners . Second, even if following the attack the community takes actions that harm the miners (e.g., exchange rate collapse), such actions harms all miners whether or not they participated in the attack. If miners are small, the tragedy of the commons can arise.

The analysis is meant to distill which assumptions are necessary to establish the protocol's security. We will find that additional assumptions are required to explain the lack of attacks on the protocol. We discuss these in Section 5.

4.1 Rental Model

We first consider a stylized economic environment in which miner's can freely enter and exit. Miners can choose the amount of computational power used for the protocol, and may flexibly adjust it (for example, by renting capacity from cloud computing service). It will be convenient to write the miner's cost as c USD for each compute of a block's hash. Miners are assumed to be profit driven. The attacker has the same capabilities and costs as any other miner.

We consider an attacker who wishes to deceive a merchant into believing that a transactions was finalized, but later the transaction is not included in the consensus chain. The merchant uses an observer node that participates in the protocol (except that it doesn't attempt to mine new blocks) to determine whether transactions were finalized. The following theorem gives the economic security of Bitcoin in this environment. The theorem posits that the net cost of a consistency attack is 0, and intuitively, this is because the block rewards on the attack fork gained by the attacker can be used to fund the attack itself.

Proposition 1. *The economic security of Bitcoin's Nakamoto in the rental model is* 0*, that is, the attacker can violate consistency and deceive a merchant at* 0 *net cost. This holds even if the*

[10] Moreover, if such a significant partition were to occur, it is likely to be known to the nodes and the network users.

merchant perfectly monitors all network activity and requires additional confirmations for finality.

The key argument is that the attacker's can create an attack chain that will get adopted as the consensus chain. When the attacker mines a block in the consensus chain, the cost and rewards of the attacker are the same as an honest miner. If honest miners find it profitable to mine blocks, so does the attacker. The proof follows similar arguments to [Budish, 2022, Auer, 2019, Moroz et al., 2020, Gans and Halaburda, 2023]. We include it here for completeness.

Proof. Suppose the transaction of interest to the attacker and merchant is in block b_n. The attacker waits for the merchant to confirm the transaction, which (by liveness) occurs at some time t. Suppose that at time t the longest chain recorded by any honest miner is $(b_0, \ldots, b_{n-1}, b_n, \ldots, b_{n+K})$.

The attacker starts mining an attack chain at time t, communicating each mined block immediately. The attacker continues to the chain $(b_0, \ldots, b_{n-1}, b'_n, \ldots, b'_{n+L})$ until for some $L > K$ when the attack chain becomes the longest chain and adopted as the consensus chain by all miners.

The attacker pays $D \cdot c$ in expectation per mined block. If the attack chain is adopted by honest miners, the attacker receives p_b per mined block. Because honest miners also pay $D \cdot c$ in expectation per mined block and are profit maximizing, it must be that $p_b \geq D \cdot c$, and the attacker's net cost is zero. Finally, the probability that the attack chain of height $n + L$ becomes the longest chain can be made arbitrarily close to 1 if the attacker computes the same number of hashes over a shorter period of the time. □

Intuitively, if the protocol cannot differentiate between the attacker and an honest miner, the protocol cannot punish the attacker.

The models of [Budish, 2022], [Auer, 2019] allow for a positive attacker's cost, because they allow for punishment due to factors outside the protocol itself (e.g., exchange rate collapse) or that the attacker has different costs from other miners. We discuss such factors next.

4.2 Bribery model and the tragedy of the commons

Consider now an economic environment in which the set of miners is fixed, and the attacker can only obtain mining power from existing miners. For example, miners may employ specialized mining equipment[11] or have access to other restricted resources (e.g., cheap electricity, computing facilities). We use a stylized economic environment to capture an extreme fixed-cost of mining, allowing the cost of setting up a miner to be prohibitively high. We refer to this economic environment as the bribery model.

In addition, we allow that an attack causes significant harm to the miners. When there is no free entry of miners it is possible that $p_b > D \cdot c$ and existing miners gain strictly positive profits from mining. The equipment miners hold is valuable because of the expectation of future profits from the protocol. If the community responds to an attack by devaluing the exchange rate or by decreasing usage, then the future block reward p_b decrease and miner's equipment decrease in value.

[11] [Garratt and van Oordt, 2020] analyze a security model with miner fixed costs. [Prat and Walter, 2021] analyze miner's investment in dedicated ASIC chips.

Miners may also hold the platform's coin or have other forms of stake in the system. We allow for any harm caused to miners from the attack and denote the collective harm to miner's from an attack by C_A.

The attacker can attack the protocol by committing to an incentive contract, which we refer to as a bribery contract [Winzer et al., 2019, Bonneau, 2016a, McCorry et al., 2018, Judmayer et al., 2021]. To focus on the role of economic incentives, we assume that miners are purely profit driven. Miners will participate in the attack if it is in their selfish best interest to do so given the payments they stand to receive from the protocol, the bribes from the miner, and the harm they may suffer due to an attack.

A bribery contract induces a game among the miners. For simplicity, we assume that miners play a simultaneous move game in which all miners observe the bribery contract and make a simultaneous choice whether to participate in the attack or not. Moreover, we assume that there is a continuum of miners, each controlling an infinitesimal computational power.

The following theorem shows that even if miners are substantially harmed by an attack, it is cheap for the attacker to incentivize individual small miners to participate in an attack.

Proposition 2. *Assume there is a continuum of miners, each controlling an infinitesimal fraction of the total mining power. Then the attacker's cost of deceiving a merchant can be arbitrarily close to* 0*.*

Proof. As in the proof of Proposition 1, the attacker waits until the merchant confirms the transaction in block b_n at time t. Only then, the attacker publishes a bribery contract that pays miners who mine the blocks $b'_n, \ldots, b'_{n+L}$ that form the attack chain $(b_0, \ldots, b_{n-1}, b'_n, \ldots, b'_{n+L})$. The contract commits to pay $\tilde{p}_b$ for each mined block.[12] The the block reward in each of the blocks $b'_n, \ldots, b'_{n+L}$ is directed to an address that is controlled by the attacker.

The bribery contract induces a game among miners. The miner may choose to attempt to mine the honest chain, mine the attack chain, or stop mining. If sufficiently many miners mine the honest chain and refuse to mine the attack chain, the attack fails. If sufficiently many miners mine the attack chain, the attack succeeds. Miners are guaranteed by the bribery contract to receive the reward $\tilde{p}_b$ regardless of whether the attack succeeds or fails, but will only collect the honest block reward p_b if the attack fails. If the attack succeeds, all miners suffer the cost of the attack c_A, where c_A denotes the normalized harm per hash to an individual miner from an attack. (recall that miners are anonymous).

The miner's payoff (per hash) is summarized in Table 1.

Because miners must mining the honest chain when there is no attack, we have that $p_b/D - c \geq 0$.

By assumption, miners are small and they are not pivotal to the success of the attack. Thus, if the attacker pays $\tilde{p}_b > p_b$ it is a dominant strategy for each individual miner to participate in the attack. The attacker's net cost per block in the attack chain is $\tilde{p}_b - p_b \approx 0$.

[12] Such a commitment can be implemented by a smart contract running on a different blockchain. To receive payment, the miner submits the block to the smart contract which is able to very it is a valid block that extends the attack chain.

	Attack Succeeds	Attack Fails
Mine attack chain	$\tilde{p}_b/D - c - c_A$	$\tilde{p}_b/D - c$
Mine honest chain	$-c - c_A$	$p_b/D - c$
Do not mine	$-c_A$	0

Table 1: Payoffs for a miner under the bribery contract.

□

Importantly, the attacker can induce miners to participate in the attack without compensating miners for the harm of the attack C_A. The game induced by the bribery contract creates a tragedy of the commons for miners. Even if C_A is large, it does not affect an individual miner's incentive to deter the attack. The attack harms both miners who participate in the attack as well as honest miners. If an actions by an individual miner do not change whether the attack succeeds, the harm c_A does not create incentives for individual miners to deter attacks.

The attack outlined in Proposition 2 does not require that the attacker can coordinate the miners on a specific equilibrium. For some values of $\tilde{p}_b$ the game induced by the bribery contract is a coordination game. [Biais et al., 2019] show that many chain selection rules are followed by miners in some equilibria. In the bribery game with a $\tilde{p}$ such that $\tilde{p}_b < p_b$ and $\tilde{p}_b/D > c$ there is an equilibrium in which all miners participate in the attack and the attacker gains $p_b - \tilde{p}_b$ per attack block. However, when $\tilde{p}_b < p_b$ there is an additional equilibrium in which all miners mine the honest chain. As in many coordination games, the ability to select a specific equilibria can be very strong.

5 What deters attacks?

As mentioned, in the rental model and the bribery model with small miners, the cost of an attack is 0 or arbitrarily close to 0. So in practice, why don't we see consistency attacks more often? The reason is that in practice, various other factors help to deter attacks. An interesting conclusion to draw from our economic analysis is that the *economic security of Bitcoin is comes not from the cost of mining, but rather, from these external deterrents* mentioned below.

1. *Community hard fork.* In practice, if an attack is identified, the community can apply forensic techniques to distinguish which is the attack fork and which is the benign fork. At this moment, the community can jointly make a decision to ignore the attack fork and go with the benign fork. In this case, the attacker may not be able to cash out the block rewards on the attack fork in time to help expense the cost of mining the attack fork (either through rental or bribery).

2. *Law enforcement.* In practice, to cash out from an attack, the attacker will need to withdraw fiat money from some exchange. However, exchanges are subject to know-your-customer

(KYC) rules that require them to know the identities of their customers. In this way, the attacker can risk legal prosecution for launching the attack.

3. *Mining power concentration.* Our earlier analysis of the bribery model assumes small players with infinitesimally small mining power, whose actions are non-pivotal to the outcome whether the attack succeeds or not. In practice, mining power is more concentrated in the form of mining farms and mining pools. For a big player, whether it cooperates in the attack may be pivotal to whether the attack is successful. For example, consider the extreme case where the attack is guaranteed to succeed if a big miner cooperates; otherwise it will fail. To bribe such a big miner, it is no longer sufficient for the attacker to only offer slightly higher than the normal block reward; it must also compensate for the c_A part of the cost to the big miner should the attack succeed (e.g., the loss suffered from currency devaluing). Therefore, although mining power concentration hurts the decentralized nature of the cryptocurrency, it actually helps enhance the economic security in this respect.

4. *Other deterrents.* In practice, there may also be other deterrents. For example, miners may not be fully profit-driven, or may be altruistic. In such cases, a bribery attack may fail due to not being able to solicit enough mining power to cooperate. In this case, the attacker cannot use the block rewards on the attack fork to help expense the cost of the attack.

In a two-sided market with free entry, the cost of mining a block equals the mining reward [Huberman et al., 2021
It is commonly believed that higher mining reward (i.e., higher cost of mining) increases the economic security of a cryptocurrency. However, our economic analysis reveals interesting subtleties regarding this claim. Specifically, our analysis suggests that in a decentralized world with small miners, *higher mining reward only increases the initial capital cost needed for an attack; it does not actually reduce the cost of the attack itself.*

6 An Economically Secure Permissionless Consensus Protocol

One might worry that the negative results in the previous sections generalize to other protocols, in that any system with many anonymous miners/ validators will suffer from the tragedy of the commons and provide miners with weak incentives to secure the system. Of course, we can design economically secure protocols that are not decentralized. But is it possible to design an economically secure protocol without losing the useful decentralization properties?[13]

In this section we propose a variation of the Nakamoto protocol that we name the Stubborn Nakamoto protocol. As in the classic Nakamoto protocol, miners in the Stubborn Nakamoto protocol are anonymous and free to enter and exit. We provide an economic security guarantee for the Stubborn Nakamoto protocol, showing that a protocol can be open and secure. Instead of

[13] E.g., can we maintain the free-entry of new miners that allows [Huberman et al., 2021] to establish their monopoly free property?

relying on incentives to provide secure, the protocol's security originates from communication and consensus.

6.1 Protocol Definition: Stubborn Nakamoto

The Stubborn Nakamoto protocol differs from the standard Nakamoto protocol by two main aspects: (i) an extra 2Δ conflict-discovery window before finalizing a block, and (ii) miners ignore any block that contradicts a finalized block. The protocol assumes synchronous communication, that is, any message sent by an honest miner reaches other honest miners within some $\Delta > 0$ time.

Description of Stubborn Nakamoto. Below, we describe our Stubborn Nakamoto protocol. We make a standard "implicit echoing" assumption, that is, every one (consensus node or observer) echos every fresh message they see. In this way, if any honest player sees a message at time t, then every honest player will have seen it by $t + \Delta$.

- All consensus nodes run Nakamoto's blockchain protocol, except for the following changes.
- All consensus nodes and observers: when a block b newly finalizes in Nakamoto (i.e., it becomes at least k blocks deep in some longest chain), wait 2Δ time. If the player has heard of another block $\mathsf{b}' \neq \mathsf{b}$ at the same height that is at least k deep, then halt; otherwise finalize the block b.
- If a node receives a block that conflicts with a finalized block, the node ignores that block and any block that extends it.

Informally speaking, because in Stubborn Nakamoto, we introduce the 2Δ conflict-discovery period before a node finalizes, consistency always holds no matter how many nodes are controlled by the adversary. Liveness of Stubborn Nakamoto can only be broken if in the underlying Nakmoto, some block b at least k-deep appears in some honest node's view in some round r; however, by the end of round $r + 2\Delta$, some other conflicting block $\mathsf{b}' \neq \mathsf{b}$ at the same height also at least k-deep appears in some honest node's view. The condition under which liveness of Stubborn Nakamoto is broken is very close to the consistency condition of the underlying Nakamoto, but not exactly implied by the consistency definition itself — this is why in the liveness proof, we need invoke lemmas used in the analysis of Nakamoto consensus [Pass et al., 2017, Shi,] in a non-blackbox manner. *Essentially, Stubborn Nakamoto transforms a consistency violation of the underlying Nakamoto into a liveness violation.*

Theorem 1 (Consistency of Stubborn Nakamoto)**.** *As long as the adversary cannot break the Δ-synchrony assumption, then, consistency is respected even in the presence of an adversary with infinite budget and unbounded computational power, under any number of corrupted players.*

Proof. Ignoring any block that conflicts with a finalized block does not affect the consistency and liveness of the underlying Nakamoto.

Henceforth, if an honest player finalizes some block b at time t, we say that $t - 2\Delta$ is the time at which the player attempts to finalize the block b. Self-consistency is by construction since the underlying Nakamoto protocol finalizes at each height only once, so every honest player will only attempt to finalize at each height once.

Suppose some honest player u finalizes a block b at height h and time t, and another honest player u' finalizes a block b' at the same height h and $t' \geq t$. This means that at $t - 2\Delta$, player u has already observed a chain denoted chain where $\mathsf{chain}[h] = \mathsf{b}$ is k or more blocks deep. By the implicit echoing assumption, at $t - \Delta$, player u' must have observed chain. There are two cases:

1. If at $t-\Delta$, u' has not observed any other chain chain' at least as long as chain, u' will attempt to finalize b at $t - \Delta$ and will not attempt to finalize any other block at height h.

2. If in some round $r \leq t - \Delta$, u' has observed some other chain chain' at least as long as chain. Then, u' must attempt to finalize some block b' at height h in round $r \leq t - \Delta$. It must be that $\mathsf{b}' = \mathsf{b}$ since otherwise u' will halt at $t-\Delta$, contradicting the fact that u' finalizes a block at height h in round $t' \geq t$.

□

Theorem 2 (Liveness of Stubborn Nakamoto)**.** *Suppose that the "network-aware honest majority" assumption holds (see Appendix A for a technical statement of the assumption); further, let the cutoff parameter k be a super-logarithmic function in some desired security parameter λ. Then, Stubborn Nakamoto satisfies consistency and liveness.*

To prove Theorem 2, we need to use some intermediate stochastic bounds used to prove the consistency and liveness of Nakamoto. We state the lemmas needed in Appendix A.3 and prove Theorem 2 in Appendix B.

Remark 1. *We point out that Theorem 1 and Theorem 2 both hold even in a network where nodes can come and go dynamically, essentially in the same model as previous works that analyze Nakamoto consensus [Garay et al., 2015, Pass et al., 2017, Shi,] — see Appendix C for more explanations. In Appendix C, we show that in a model with dynamic joins and leaves, it is impossible to simultaneously attain infinite economic security (for consistency) and liveness under corrupt majority, too.*

References

[wea,] Weak subjectivity. `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/weak-sub`

[zen,] Zencash statement on double spend attack. `https://blog.horizen.global/zencash-statement-on-doub`

[Auer, 2019] Auer, R. (2019). Beyond the doomsday economics of'proof-of-work'in cryptocurrencies.

[Biais et al., 2019] Biais, B., Bisiere, C., Bouvard, M., and Casamatta, C. (2019). The blockchain folk theorem. *The Review of Financial Studies*, 32(5):1662–1715.

[Bonneau, 2016a] Bonneau, J. (2016a). Why buy when you can rent? - bribery attacks on bitcoin-style consensus. In *Financial Cryptography Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 19–26. Springer.

[Bonneau, 2016b] Bonneau, J. (2016b). Why buy when you can rent? bribery attacks on bitcoin-style consensus. In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer.

[Budish, 2022] Budish, E. B. (2022). The economic limits of bitcoin and anonymous, decentralized trust on the blockchain. *University of Chicago, Becker Friedman Institute for Economics Working Paper*, (83).

[Buterin and Zamfir, 2015] Buterin, V. and Zamfir, V. (2015). Casper. `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`.

[Chiu and Koeppl, 2022] Chiu, J. and Koeppl, T. V. (2022). The economics of cryptocurrency: Bitcoin and beyond. *Canadian Journal of Economics/Revue canadienne d'économique*, 55(4):1762–1798.

[Daian et al., 2019] Daian, P., Pass, R., and Shi, E. (2019). Snow white: Provably secure proofs of stake. In *FC*.

[Gans and Halaburda, 2023] Gans, J. S. and Halaburda, H. (2023). " zero cost"majority attacks on permissionless blockchains. Technical report, National Bureau of Economic Research.

[Garay et al., 2015] Garay, J., Kiayias, A., and Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer.

[Garratt and van Oordt, 2020] Garratt, R. and van Oordt, M. R. (2020). Why fixed costs matter for proof-of-work based cryptocurrencies. *Available at SSRN 3572400*.

[Guo and Ren, 2022] Guo, D. and Ren, L. (2022). Bitcoin's latency–security analysis made simple. *arXiv preprint arXiv:2203.06357*.

[Huberman et al., 2021] Huberman, G., Leshno, J. D., and Moallemi, C. (2021). Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *The Review of Economic Studies*, 88(6):3011–3040.

[Judmayer et al., 2021] Judmayer, A., Stifter, N., Zamyatin, A., Tsabary, I., Eyal, I., Gazi, P., Meiklejohn, S., and Weippl, E. (2021). Pay to win: Cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on pow cryptocurrencies. In *FC WTSC*.

[Karakostas and Kiayias, 2021] Karakostas, D. and Kiayias, A. (2021). Securing proof-of-work ledgers via checkpointing. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*.

[Kelso,] Kelso, C. E. Bitcoin gold hacked for $18 million. `https://news.bitcoin.com/bitcoin-gold-hacked-for-18-million/`.

[McCorry et al., 2018] McCorry, P., Hicks, A., and Meiklejohn, S. (2018). Smart contracts for bribing miners. In *Financial Cryptography Workshops*, volume 10958 of *Lecture Notes in Computer Science*, pages 3–18. Springer.

[McKinsey and Company, 2021] McKinsey and Company (2021). The 2022 mckinsey global payments report.

[Moroz et al., 2020] Moroz, D. J., Aronoff, D. J., Narula, N., and Parkes, D. C. (2020). Double-spend counterattacks: Threat of retaliation in proof-of-work systems. *arXiv preprint arXiv:2002.10736*.

[Nakamoto, 2009] Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. `http://bitcoin.org/bitcoin.pdf`.

[Nesbitt,] Nesbitt, M. Deep chain reorganization detected on ethereum classic (etc). `https://blog.coinbase.com/ethereum-classic-etc-is-currently-being-51-attacked-33be13ce32de`.

[Neu et al., 2021] Neu, J., Tas, E. N., and Tse, D. (2021). Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*.

[Neu et al., 2022] Neu, J., Tas, E. N., and Tse, D. (2022). The availability-accountability dilemma and its resolution via accountability gadgets. In *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, volume 13411 of *Lecture Notes in Computer Science*, pages 541–559. Springer.

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

[Prat and Walter, 2021] Prat, J. and Walter, B. (2021). An equilibrium model of the market for bitcoin mining. *Journal of Political Economy*, 129(8):2415–2452.

[Sheng et al., 2021] Sheng, P., Wang, G., Nayak, K., Kannan, S., and Viswanath, P. (2021). BFT protocol forensics. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1722–1743. ACM.

[Shi,] Shi, E. Foundations of distributed consensus and blockchains. Textbook, https://www.distributedconsensus.net/.

[Winzer et al., 2019] Winzer, F., Herd, B., and Faust, S. (2019). Temporary censorship attacks in the presence of rational miners. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 357–366.

A Formal Definitions: Distributed Consensus and Execution Model

To formally state the conditions under which Nakamoto has provable guarantees, we introduce some notations regarding the protocol's execution model.

We assume a synchronous network with a maximum message delay of Δ. This means that if an honest node i sends a message to an honest node j in round r, the message is guaranteed to be delivered in or before round $r + \Delta$. We assume that there are n nodes participating in the consensus protocol, all with equal mining power (since a node with larger mining power can simply be viewed as multiple nodes). Let p be the probability that a single node finds a valid block in a round, where the probability p is related to the mining difficulty parameter. We use the notation $\rho \in [0, 1)$ to denote the fraction of nodes controlled by the adversary.

A.1 Distributed Consensus, Consistency and Liveness

We review what is a distributed consensus protocol (specifically, a blockchain protocol), and the consistency and liveness properties. A distributed consensus protocol is parametrized with some security parameter denoted λ. Typically, we want the probability of security failure to be negligibly small in the security parameter λ. In other words, the security failure probability should be upper bounded by any inverse polynomial function in λ. To achieve this, we need to parameterize the "lookback" parameter k in a way that depends on the security parameter λ. More specifically, to get negligibly small in λ failure probability, we should k to be a super-logarithmic function in λ — this is because the security failure probability is exponentially small in k.

In a distributed consensus protocol, each node maintains a finalized linearly ordered log over time. We require that the length of each node's log is a non-decreasing function over time. We may use the notation chain_i^r to denote node i's log at the end of round r. Consistency and liveness require that there exists a negligible function $\mathsf{negl}(\cdot)$, such that with $1 - \mathsf{negl}(\lambda)$ probability,

- **Consistency:** for any honest nodes i and j (where i and j may be the same or different), and for any round numbers t and r, it must be that $\mathsf{chain}_i^r \preceq \mathsf{chain}_j^t$ or $\mathsf{chain}_i^r \succeq \mathsf{chain}_j^t$. Here, $\mathsf{chain} \preceq \mathsf{chain}$ means that the former chain is a prefix of chain' or they are the same.
- T_{conf}**-Liveness:** Liveness is parameterized by a parameter $T_{\text{conf}} = T_{\text{conf}}(\lambda, \Delta, n, \rho)$ called the confirmation time, where the confirmation time may depend on other parameters of the execution model. We want that if an honest node posts some transaction tx in some round r, then by the end of round $r + T_{\text{conf}}$, all honest nodes' finalized logs must include tx.

A.2 The "Network-Aware Honest Majority" Assumption

Pass et al. [Pass et al., 2017] and Shi's textbook [Shi,] showed that Nakamoto's protocol satisfies consistency and liveness under the following parameter constraints:

1. $\nu = 2pn\Delta < 0.5$;
2. there exists some an arbitrarily small constant $\phi \in (0, 1)$ such that $(1-\rho)(1-\nu) \geq (1+\phi)\rho$. In particular, the term $(1-\rho)(1-\nu)$ can be viewed as the fraction of honest mining power discounted by the network delay Δ, and therefore this condition essentially requires that the honest mining power, even when discounted the network delay, exceeds corrupt mining power by a small constant margin.

A.3 Useful Lemmas

The following useful stochastic bounds were proved in earlier works [Pass et al., 2017, Shi,]. As mentioned earlier, T is a *convergence opportunity* if a single honest block is mined in round T and no honest blocks are mined in the surrounding Δ rounds before and after.

Lemma A.1 (Lower bound on convergence opportunities [Pass et al., 2017, Shi,])**.** *For any positive constant η and any k that is a super-logarithmic function in λ, except with negligibly small in λ probability over the choice of the execution, the following holds: for any $t_0, t_1 \geq 0$ such that $t := t_1 - t_0 > k/\alpha$, we have that*

$$C[t_0 : t_1] > (1-\eta)(1-2pn\Delta)\alpha t$$

where $\alpha := p(1-\rho)n$ denotes the expected number of honest nodes that mine a block in each round.

Lemma A.2 (Upper bound on adversarially mined blocks [Pass et al., 2017, Shi,])**.** *For any constant $0 < \epsilon < 1$, for any k that is a super-logarithmic function in λ, except with negligibly small in λ probability over the choice of the execution, the following holds: for any $t \geq k/\beta$, the number of adversarially mined blocks in any t-sized window is upper bounded by $(1+\epsilon)\beta t$.*

Lemma A.3 (Total block upper bound [Pass et al., 2017, Shi,])**.** *For any positive constant ϵ and any k that is a super-logarithmic function in λ, except with negligibly small in λ probability over the choice of the execution, the following holds: for any r and t such that $np(t-r) \geq k$, the total number of blocks successfully mined during $(r, t]$ by all nodes (honest and corrupt alike) is upper bounded by $(1+\epsilon)np(t-r)$.*

B Proof of Theorem 2

First, we claim that pruning any block (and blocks extending it) that conflicts with a finalized block does not break the consistency and liveness of the underlying Nakamoto. Specifically, in

Nakamoto, at any point of them, an honest node only needs to know what are the longest chains in its records — it always finalizes a prefix of some longest chain and mines off some longest chain. Consistency of Nakamoto guarantees that except with negligible probability, if block B^* is finalized in some honest node's view, then any chain that conflicts with B^* will never become the longest chain in any honest node's view. In other words, if honest nodes do not save chains that conflict with B^*, it will not affect the protocol's behavior.

The liveness of Stubborn Nakamoto can only be broken some block b that is at least k blocks deep is in honest view at time r, and a different block $\mathsf{b}' \neq \mathsf{b}$ at the same height as b also at least k blocks deep is in honest view at time $r + 2\Delta$. We show that this happens with only negligible in λ probability.

Define convergence opportunity in the same way as Section 17.3 of Shi [Shi,]. Specifically, T is a convergence opportunity if a single honest block is mined in round T and no honest blocks are mined in the surrounding Δ rounds before and after. Let chain be the chain in honest view in round r containing b at least k-deep, and let chain' be the chain in honest view in round $r + 2\Delta$ containing b' at least k-deep. Let $\mathsf{b}_{\text{common}}$ be the last honest block shared in common for chain and chain'. Suppose $\mathsf{b}_{\text{common}}$ is mined in round $s - 1$. It holds that all blocks in chain and chain' after $\mathsf{b}_{\text{common}}$ is mined in round s or later. Let $\tau = r - s$. By total block upper bound (see Lemma A.3 of Appendix A.3), it must be that $\tau > \frac{k}{2pn}$.

Henceforth, let $\mathbf{C}[t : t']$ denote the number of convergence opportunities between t and t', and let $\mathbf{A}[t : t']$ denote the number of adversarial blocks mined between t and t'. To cause chain and chain' to fork off at the end, it must be that $\mathbf{C}[s : r - \Delta] \leq \mathbf{A}[s : r + 2\Delta]$, since otherwise, there must be an honest block B mined during a convergence opportunity between $[s, r - \Delta]$ and B must appear in both chain and chain'. Therefore, it suffices to prove that except with negligible in λ probability, it must be that $\mathbf{C}[s : r - \Delta] > \mathbf{A}[s : r + 2\Delta]$.

By the lower bound on convergence opportunity (see Lemma A.1 of Appendix A.3), the following holds except with negligible probability:

$$\mathbf{C}[s : r - \Delta] > (1 - \epsilon)(1 - 2pn\Delta)\alpha(\tau - \Delta)$$

where $\alpha = p(1-\rho)n$ denotes the expected number of honest nodes that mine a block in each round, and ϵ is an arbitrarily small constant.

By adversarial block upper bound (see Lemma A.2 of Appendix A.3),

$$\mathbf{A}[s : r + 2\Delta] > (1 + \epsilon')\beta(\tau + 2\Delta)$$

where ϵ' is an arbitrarily small constant, and $\beta = p \cdot \rho n$ denotes the expected number of corrupt nodes that mine a block in each round.

Thus for any positive constant ϕ, as long as $0 < 2pn\Delta < 0.5$, there exist sufficiently small constants ϵ, ϵ', and ϵ_1 such that the following holds for sufficiently large k:

$$\begin{aligned}
\mathbf{C}[s : r - \Delta] &> (1-\epsilon)(1-\nu)\alpha(\tau - \Delta) \\
&> (1-\epsilon_1)(1-\nu)\alpha\tau \\
&> (1-\epsilon_1)(1+\phi)\beta\tau \\
&> (1+\epsilon')(1+\frac{\phi}{2})\beta\tau \\
&> (1+\epsilon')\beta(\tau + 2\Delta)
\end{aligned}$$

where the last inequality holds for sufficiently large k because $\alpha\Delta = O(1)$ and $\alpha\tau = \Theta(k)$.

C Dynamic Joins and Leaves

Earlier in Remark 1, we pointed out that our main theorems for Stubborn Nakamoto, that is, Theorem 1 and Theorem 2 hold even in a model with dynamic joins and leaves — essentially the same model as earlier works that analyze Nakamoto's protocol [Garay et al., 2015, Pass et al., 2017, Shi,]. Specifically, we may assume that if a node joins late in some round r, then in each round $r' \geq r$ before it drops offline again, it will have received all messages sent by honest nodes in round $r' - \Delta$ or earlier. Further, when a node newly joins, it may receive arbitrary messages from adversarial nodes. In our Stubborn Nakamoto, a node has to be online for at least 2Δ rounds to finalize a block. More specifically, we can account for dynamic joins and leaves in the consistency proof as follows. We can simply replace every occurrence of $t - \Delta$ with $\max(t - \Delta, \mathsf{join}(u'))$ where $\mathsf{join}(u')$ denotes the last time u' joins before t'. The liveness proof still holds despite the dynamic joins and leaves because all the lemmas we rely on (see Appendix A.3) hold in a model with dynamic joins and leaves.

Below, we show that in such a model with dynamic joins and leaves, it is not possible to simultaneously achieve consistency and liveness both under corrupt majority. This means that if we want to ensure liveness under corrupt majority, then as long as the attacker can pay the cost to corrupt enough nodes, it can break consistency. In other words, a protocol that enjoys liveness under corrupt majority cannot enjoy infinite economic security.

Theorem 3. *In a model with unauthenticated channels allowing dynamic joins and leaves, it is impossible to achieve consistency and liveness both under corrupt majority in mining power. Moreover, this impossibility holds even in a proof-of-work model.*

Proof. Imagine a scenario with n nodes and all nodes have equal mining power. Assume that n is an even number, and suppose half of them are corrupt. Let H be the set of honest nodes, and let C be the set of corrupt nodes. Initially, suppose the corrupt nodes do not send messages to honest nodes, and they ignore all messages from honest nodes. Therefore, the corrupt nodes basically run the honest consensus protocol amongst themselves. In other words, there are two separate executions, the honest execution and the corrupt execution. Suppose that in both the honest and

corrupt executions, transactions are randomly sampled from the same distribution $\mathcal{D}$ and posted to the honest (or corrupt) nodes.

At some time t, in both the honest execution and corrupt execution, some nodes will have finalized some transactions. Since the corrupt nodes sample their own random transactions, with high probability, the two executions' finalized logs differ in the first confirmed transaction.

Suppose at some time $t' > t$, one honest node drops out, one corrupt node drops out, and they are replaced by a newly joining honest node u and a newly joining corrupt node. The corrupt nodes now forward to u the messages they withheld from the honest nodes earlier. At this moment, u observes two independent executions that are identically distributed. By liveness, it must finalize some log. By consistency, the finalized log must be consistent with the honest nodes'. However, since the distributions of the two executions are identical from u's perspective, it is not possible for u' to choose the correct log to finalize with probability more than $1/2$. □