On the Viability of Open-Source Financial Rails: Economic Security of Permissionless Consensus

Jacob D. Leshno* Rafael Pass† Elaine Shi‡

March 19, 2025§

Abstract

Bitcoin demonstrated the possibility of a financial rail with a novel economic structure that circumvents monopoly and antitrust concerns. However, questions remain as to whether this open structure can satisfy the security requirements of mainstream financial applications. This paper investigates the underlying consensus protocols to determine whether they can deliver a financial rail that is both open and secure.

We answer affirmatively by constructing a protocol that provides an economically meaningful security guarantee while also preserving Bitcoin's permissionless and open design. Notably, our protocol's security does not rely on monetary payments to miners or high energy consumption, which our analysis indicates are ineffective. Our framework integrates economic theory with distributed systems theory and formalizes the role of the user community.

*University of Chicago Booth School of Business, Jacob.Leshno@ChicagoBooth.edu.

†Tel Aviv University and Cornell Tech, Computer Science, rafaelp@tau.ac.il.

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

§This work is supported by NSF grant number 2212747 and by the Robert H. Topel Faculty Research Fund at the University of Chicago Booth School of Business. We thank Srivatsan Sridhar and Dionyziz Zindros for their insightful and detailed comments. We thank Ittai Abraham, Garth Baughman, Dirk Bergemann, Eric Budish, Francesca Carapella, Alex Frankel, Rod Garratt, Niels Gormsen, Emir Kamenica, Anil Kashyap, Andrew Lewis-Pye, Yueran Ma, Dahlia Malkhi, Harry Pei, Tim Roughgarden, Marciano Siniscalchi, Ertem Nusret Tas, David Tse, Maarten van Oordt, Luigi Zingales, and seminar participants at the ACM SIGecom Winter Meeting, the Federal Reserve Board of Governors, Market Design NBER, a16z Research Lab, Stanford, Princeton, Notre Dame, the Northwestern Conference on the Interface between Computer Science and Economics, the London Symposium on Information Theory, and the Simons Institute for the Theory of Computing. All errors are our own.

1 Introduction

A long-standing literature has demonstrated the critical role of the financial system in fostering economic growth (Schumpeter, 1911, Rajan and Zingales, 1998), and the necessity of a trusted financial infrastructure that safeguards against fraud and deception (La Porta et al., 2006, Guiso et al., 2004, 2008). At the same time, economists have expressed concerns about inefficiencies and rent-seeking behavior within the financial system (Greenwood and Scharfstein, 2013, Admati and Hellwig, 2014, Philippon, 2015).

These concerns are particularly pronounced in financial rails – the record-keeping and payment services that underpin all financial activity. Firms operating these foundational services benefit from substantial network effects and economies of scale, enabling them to extract significant rents (Rosenbaum et al., 2017, Morningstar, 2024, McKinsey & Company, 2024). In response, regulators have undertaken regulatory action and high-profile antitrust litigation, including a recent lawsuit by the U.S. Department of Justice accusing Visa of monopolizing the debit card market (The New York Times, 2024).

The emergence of Bitcoin presented a new kind of financial rail that averts these antitrust concerns. Bitcoin offers its users similar record-keeping and payment services, but instead of being controlled by a firm, Bitcoin is governed by the open-source Nakamoto consensus protocol. The Nakamoto protocol is permissionless, meaning that Bitcoin is operated by anonymous, interchangeable computer servers (called nodes or miners) that can freely enter or exit. Bitcoin's open economic structure – in particular, free entry of its operators – prevents any participant from exercising monopoly power (Huberman et al., 2021). This novel economic structure, made possible by permissionless consensus protocols, has attracted significant public attention and spurred extensive research. It inspired the Web3 movement, a vision for the next generation of the internet that seeks to leverage this technology to curtail Big Tech's market power and develop open-source alternatives.[1]

But can a financial rail be both open and secure? Security concerns[2] cast doubt on the viability of permissionless consensus protocols for mainstream financial applications (e.g., Budish 2025). There are fundamental concerns about whether this open technology can provide the security needed for mainstream applications: because the system is open, an

[1] Dixon (2024) offers development of these ideas, highlighting precedents such as the success of the open-source operating system Linux and the community-controlled Domain Name System. See also Wired (2022) and Wired (2019).

[2] There are numerous additional challenges to adopting this technology as a mainstream financial rail. This paper centers on the security concerns that are fundamental to the technology and paramount for its adoption. We discuss additional challenges in Section 6.

attacker can join as an operator and compromise it from within. For example, Bitcoin's Nakamoto protocol is vulnerable to corrupt-majority attacks (often called 51% attacks or double-spending), in which an attacker controlling most of the system's operators can revert transactions. Although Bitcoin itself has not been attacked, multiple other systems running the Nakamoto protocol have experienced such attacks (Shanaev et al., 2019). This vulnerability has drawn extensive attention from practitioners, economists, and computer scientists – raising the question of whether any permissionless consensus protocol can be secured at a reasonable cost.

This paper addresses this economic design problem: is it possible to design a financial rail that is both open and secure? We draw on distributed systems theory to characterize the system's desired functionality and key technological challenges, enabling us to evaluate general protocol designs and define economically meaningful properties. We introduce a definition of *economic security* that captures whether users remain protected against fraud or deception, allowing for general protocol designs and general attacks. To capture free entry[3] of the system's operators, we adopt the distributed systems definition of permissionless consensus protocols (Pass and Shi, 2017a, Lewis-Pye and Roughgarden, 2023).

Our primary finding is that a permissionless and economically secure protocol is indeed feasible. We demonstrate this feasibility by constructing a variation of Bitcoin's Nakamoto protocol that is economically secure and maintains its permissionless design. Notably, our protocol adopts a different approach to security; its security does not depend on monetary incentives for miners or their excessive energy consumption, which we identify as ineffective.

To perform our analysis, we use a framework that integrates economic theory with distributed systems theory, both components being necessary and complementary. We use distributed systems theory to describe the precise functionality a financial rail must deliver: maintaining agreement on a ledger (see Section 2). A distributed systems model is essential for analyzing general protocol design, general user behaviors, and general attacks. Additionally, economic incentives depend on the system's ability to detect and punish deviations from prescribed behaviors. The distributed systems model enables us to formally capture the knowledge of each participant and determine who is able to detect an attack. An economic

[3]The novel economic structure enabled by permissionless consensus protocols is often described as "decentralized," loosely indicating that no central authority controls the system. There are numerous properties that formalize different notions of decentralization. Our analysis focuses on the free-entry property for two reasons. First, it is a precisely defined property that offers a valuable guarantee to users: as shown by Huberman et al. (2021), large miners cannot profitably manipulate transaction fees when there is free entry of miners. Second, free entry lies at the core of many notions of "decentralization," since controlling who may operate the system effectively grants a measure of control over the system itself.

model is essential to capture the challenges faced by a permissionless protocol operated by profit-motivated participants. Our analysis must account for the endogenous composition of nodes, including the possibility that an attacker may control multiple nodes to exploit system vulnerabilities.[4] The economic model allows us to account for attacks that operate or incentivize nodes within the system, and it provides a meaningful quantification of such security vulnerabilities in terms of the cost of the attack. In addition, our framework allows us to consider dependencies on external factors such as the community of users, and, in particular, to evaluate the community's capabilities to detect and respond to attacks.

Our framework allows us to provide a definition of *economic security* for general consensus protocols that captures whether users of the system are protected against deception or fraud in the face of general attacks. To illustrate the definition, consider a merchant who wishes to receive payment. The merchant wants to ensure that the payment is valid, i.e., the credit is recognized by others and can be spent. While a merchant may use a payment terminal to verify a transaction, the merchant can only receive funds from the bank if the bank's terminal has also confirmed the payment. A key challenge for digital payment systems is ensuring agreement among participants. For instance, the system must prevent situations where the merchant's terminal indicates a payment was completed while the bank's computers have no record of that payment. If the distributed systems property *consistency* is satisfied, all participants observe the same ledger. Consequently, the merchant's terminal is guaranteed to be in agreement with the rest of the network, ensuring the merchant cannot be deceived.[5]

Because we focus on an open economic environment, we adopt an economic model of the environment and quantify an attacker's difficulty by measuring the cost of the attack. Accordingly, we define economic security as the economic cost required to violate consistency (see Section 2 for further detail). This definition captures the corrupt majority (double-spending) attack mentioned above, as well as other attacks.[6]

[4]This is in contrast to traditional distributed systems applications, for example, a firm that manages multiple data centers and wishes all data centers to hold identical replicas of the data. Traditional protocols (e.g., Paxos or Raft) were designed for such settings and function correctly under the assumption that the majority of nodes in the network operate correctly. This assumption is plausible in these traditional settings, where nodes can deviate from the protocol because of hardware failures, but large-scale simultaneous malfunctions are highly unlikely. However, such assumptions offer little protection in our setting in which nodes endogenously choose their behavior.

[5]This paper focuses on the technology's ability to offer a record-keeping service, that is, to maintain a ledger, as record-keeping is a core service of any financial system, and payment systems in particular (famously, Kocherlakota (1998) reasons that "Money is Memory"). We demonstrate in Appendix A that this record-keeping technology, combined with well-known cryptographic tools, is sufficient to construct a payment system. For a discussion of additional challenges and potential limitations, see Section 6.

[6]For example, an attacker can potentially profit from rewriting old parts of the ledger to "retroactively"

To highlight challenges in securing permissionless protocols, we analyze Bitcoin's Nakamoto protocol. We evaluate the economic security of Bitcoin's Nakamoto protocol in two economic environments, selected to underscore the deficiencies in incentives for securing permissionless protocols. Two shortcomings compromise the system's security, regardless of the level of mining rewards. First, the protocol fails to detect attacks. Malicious miners can generate an alternative ledger and cause the system to adopt it, deleting transactions and violating consistency. The protocol does not recognize that this alternative ledger is a product of malicious behavior, treating these miners the same as honest ones (we explain the protocol's rationale in Section 3.3). In particular, the protocol fails to punish such attackers and fails to deliver incentives that deter attacks. Second, even if attacks prompt an external response, such as a collapse in the exchange rate, the resulting harm affects all miners, irrespective of their involvement in the attack. If miners are small, the tragedy of the commons can arise. Together, these deficiencies underscore fundamental challenges in relying on monetary incentives to secure permissionless protocols.

But can a permissionless protocol be secured by means other than monetary incentives? The apparent lack of successful attacks on Bitcoin and Ethereum suggests that additional deterrents may be at play. We highlight one in particular: the community response. Recall that the protocol serves as a record keeper maintaining a ledger. An attack could lead nodes to adopt a corrupted version of this ledger. However, if the broader community detects the corruption, it can override the protocol's output and restore the legitimate ledger.[7] This community response is powerful, as it effectively nullifies an attack and serves as a deterrent against future breaches. That said, implementing this community response may be challenging, as it requires the community to both detect attacks and reach agreement on the correct ledger.

Our design approach is guided by the demonstrated efficacy of the community's response, which offers an alternative method of securing the protocol. Rather than relying solely on incentives and monetary rewards, a protocol could achieve security by automating and integrating a community-like response into its design. That is, the protocol should maintain agreement on the correct ledger. Instead of allowing attacks to corrupt the ledger and subsequently restoring it, the protocol should enable all nodes to preserve the correct state and

buy an option after its expiration. See Appendix D for further detail.

[7] Closely related notions are widely discussed by practitioners. For example, Buterin (2023) refers to the notion of social consensus and writes: "And if the cryptoeconomic consensus fails, whether due to a bug or an intentional 51% attack, a vast community of many thousands of developers and many more users are watching carefully to make sure the chain recovers correctly."

ignore attacks that attempt to corrupt it. By considering a version of the community response within a formal protocol, we can develop alternative protocols and rigorously evaluate whether the community response is feasible.

We demonstrate the feasibility of a permissionless consensus protocol that is economically secure by constructing such a protocol. Our protocol, which we name Stubborn Nakamoto, is a modification of Bitcoin's Nakamoto protocol. It is designed for the same environment as the Nakamoto protocol (namely, synchrony with late joiners, see Section 2 for details). In the absence of an attack, the Stubborn Nakamoto protocol and the Nakamoto protocol operate essentially identically. In particular, Stubborn Nakamoto preserves the free entry of nodes. All nodes serve equivalent roles, remain anonymous, and can join or leave at any time. The protocols differ in their response to attacks. We prove that the Stubborn Nakamoto protocol is economically secure, showing that it guarantees consistency even against an attacker with unlimited computational power and economic resources.[8] In particular, corrupt majority attacks (such as double-spending) are rendered impossible under the Stubborn Nakamoto protocol.

To achieve this economic security, we design the Stubborn Nakamoto protocol to differ from the standard Nakamoto protocol in two key aspects. First, Stubborn Nakamoto specifies an explicit condition under which a node finalizes a record in its ledger. Once a record is finalized, the node never reverts it. Second, the Stubborn Nakamoto protocol instructs nodes to wait before finalizing a record. This delay allows nodes to determine whether any other node observes an inconsistent ledger and ensures that the ledger is updated only when the node is certain that no conflict is possible. Together, these measures ensure that when a node finalizes a record, the node can guarantee that all other nodes agree on that record.

An attack can temporarily halt the Stubborn Nakamoto protocol, but the protocol can recover from attacks given external help from the community. We formally model the dependence on the community as a recovery oracle, show it can be feasibly implemented, and prove that such external dependence is necessary. Our work builds upon previous impossibility results that reveal trade-offs in protocol design and highlight the importance of modeling assumptions (including Pass and Shi (2017a), Lewis-Pye and Roughgarden (2023), Sridhar et al. (2024), see Section 4.4). The definition of the recovery oracle permits a formal treatment of the community's role, allowing us to explore appropriate assumptions given the economic environment.

[8] We maintain the standard assumption that an attacker cannot break cryptographic primitives but allow the attacker to have an unlimited hash rate.

Our analysis provides insights into the class of Proof-of-Stake (PoS) protocols. These protocols leverage cryptographic identifiers, which resolve some—but not all—of the challenges discussed above. We demonstrate how PoS protocols can strengthen security guarantees for users operating under less reliable communication. However, even such protocols must depend on the community to recover from certain attacks.

Our results show the feasibility of an economically secure financial rail that operates without the need for a trusted record keeper. It protects its users from monopoly's harm, allows free entry and exit of its operators, and renders its operators interchangeable. The analysis offers an alternative explanation for Bitcoin's security, suggesting a broader perspective that includes the role of the community. Bitcoin's Nakamoto and our Stubborn Nakamoto differ only under an attack, and one interpretation of our work is that analysis of protocols under attack cannot take a narrow view that only considers the formal protocol description. In particular, attacks that deceive the protocol may be rendered ineffective if they can be detected and reversed by the community. The security of a ledger, therefore, may have little to do with the amount of resources spent on it. Instead, security is ensured by establishing agreement among all participants.

1.1 Related work

Our work is motivated by antitrust challenges in financial rails, particularly within the payment-processing market. McKinsey & Company (2024) estimate the revenue of the global payments industry to be $2.4 trillion in 2023. Rosenbaum et al. (2017), Morningstar (2024) argue payment firms benefit from market power that is supported by network effects and economies of scale, leading to limited competition and significant barriers to entry. Congressional testimony by Aaron Klein (2020) highlights how underserved and disadvantaged populations are impacted by payment providers. For lists of antitrust lawsuits and regulatory actions, see Table 5 in Herkenhoff and Raveendranathan (2020) and Hayashi and Maniff (2019), respectively. Recent high-profile antitrust lawsuits include the U.S. vs. American Express (DOJ, 2010), the European Commission imposing interchange fee regulation (European Parliament and Council, 2015), and the DOJ blocking Visa's acquisition of Plaid (DOJ, 2020).

This paper builds on work in economics and in distributed systems, bringing together elements of both.

We build on a rich literature in economics that studies double-spend attacks on Bitcoin's Nakamoto protocol, including Budish (2025), Auer (2019), Bonneau (2016), Chiu and Koeppl

(2022), Moroz et al. (2020), Garratt and van Oordt (2023), Pagnotta (2022), and Gans and Halaburda (2024). This literature focuses on the double-spend attack against Bitcoin's Nakamoto protocol. Our framework allows us to give a definition of economic security that considers general attacks and to analyze alternative protocol designs. Abraham et al. (2006), Abadi and Brunnermeier (2022), Garay et al. (2013), Biais et al. (2019), Badertscher et al. (2021), and Halaburda et al. (2021) offer economic models of consensus protocols. See Halaburda et al. (2022) for a review.

Eyal and Sirer (2018) demonstrate that Bitcoin miners can earn higher rewards through strategic behavior. A significant body of literature followed exploring strategic actions in blockchain protocols, including Kiayias et al. (2016), Carlsten et al. (2016), Cai et al. (2024). Such manipulations do not compromise the protocol's consistency,[9] and are therefore beyond of the scope this paper.

Our work builds on the seminal work in distributed systems. We adopt the modeling approach established in this literature in works such as Pease et al. (1980), Lamport et al. (1982), Dwork et al. (1988), Fischer et al. (1985). Our work also builds on more recent developments in distributed systems that provide rigorous analysis of the Nakamoto protocol (Garay et al., 2015, Pass et al., 2017, Guo and Ren, 2022).

We diverge from this literature in considering an economic notion of security and free entry of nodes. Classic consensus protocols, such as Paxos (Lamport, 1998), are designed for consensus among a fixed set of known nodes (e.g., a firm's server and its backups). Their goal is to provide resilience to faults such as hardware or software failure and electricity outages. In such a closed deployment scenario, incentives are a non-issue. These protocols can lose consistency if a sufficient number of nodes are corrupted, as such events are rare in the settings they were designed for. By contrast, in environments with free entry, the number of corrupt nodes is endogenous; an attacker willing to bear the cost can control as many nodes as necessary to execute an attack. Therefore, our Stubborn Nakamoto protocol is designed to remain secure regardless of the number of corrupt nodes.

Free entry imposes an additional challenge: nodes may join the protocol at any point in time. There are classic consensus protocols, such as Dolev–Strong (Dolev and Strong, 1983), that are designed for a fixed set of nodes and can operate securely (i.e., guarantee consistency and liveness) given any number of corrupt nodes. However, if nodes can join at any point and there are sufficiently many corrupt nodes, then any protocol, including Dolev–Strong,

[9]Because the Nakamoto protocol guarantees consistency under honest majority, deviation by small miners cannot lead to consistency violations.

can lose consistency. Our research builds on the works of Lewis-Pye and Roughgarden (2023), Budish et al. (2024), Pass and Shi (2017a), and Sridhar et al. (2024) that prove this impossibility result and show additional difficulties that arise in the permissionless setting.

The analysis of bribery attacks on Bitcoin builds upon the ideas introduced in Bonneau (2016). Contemporaneously and independently of this work, Newman (2023) also argues that bribery attacks are cheap when miners are small and rational because small miners are unlikely to be pivotal to an attack's success.

Earlier works suggested using irreversible "checkpoints" to secure blockchain protocols (Daian et al., 2019, Buterin and Zamfir, 2015, Karakostas and Kiayias, 2021, Buterin and Griffith, 2017, Neu et al., 2021, Ethereum Foundation, 2023, Sankagiri et al., 2021). In contrast to previous approaches that rely on adding finality gadgets or beacons that are external to the protocol, our Stubborn Nakamoto directs nodes to be "stubborn" given their local record, and it is provably secure without relying on any external support.

While we focus on protocols that allow fully anonymous nodes (commonly called proof-of-work), 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 implicates them. Such cryptographic forensic evidence can serve as a deterrent to attacks on these proof-of-stake protocols. However, Budish et al. (2024) show the limitations of this approach.

Finally, this work is related to Sridhar et al. (2023), which is concurrent with an earlier draft of the present paper. Sridhar et al. (2023) demonstrate that a range of proof-of-stake consensus protocols can be modified to guarantee consistency even under a corrupt majority and suggest a recovery approach following attacks. A key difference is that our Stubborn Nakamoto protocol does not require the setup or infrastructure of proof-of-stake protocols.

1.2 Paper outline

The remainder of the paper is organized as follows. Section 2 provides our model and motivates our definition of economic security. Section 3 analyzes Nakamoto's economic security and shows deficiencies in its incentives. We provide an explanation for the lack of attacks on Bitcoin in Section 3.5 and discuss other factors that deter attacks in practice in Appendix E. Section 4 presents the Stubborn Nakamoto protocol and proves its security. We provide comments on proof-of-stake protocols in Section 5. Section 6 concludes.

Technical details and formal definitions are in Appendix B. Omitted proofs can be found in Appendix C. Appendix D provides examples of profit from consistency attacks.

2 Model and Definitions

This section describes our model and defines economic security for permissionless protocols. The model abstracts away from some technical details, which are given in Appendix B.

2.1 Background: Distributed ledgers and consensus

A consensus protocol coordinates a collection of parties to collectively maintain a single write-only ledger.[10] We interchangeably refer to the parties who maintain the ledger as *miners* or *nodes*. Each node $i \in N$ keeps a local copy of the ledger. Denote the local ledger of node i at time t by Log_i^t. With a slight abuse of notation, we write $\mathsf{Log} \subseteq \mathsf{Log}'$ if the ledger Log' can be generated by appending data to Log. We refer to a generic piece of data in the ledger as a transaction tx, and assume that users continuously generate unforgeable[11] transactions and broadcast them to all nodes. Users may issue conflicting transactions, and the ledger needs to resolve such conflicts.[12]

We say that node i finalizes a transaction tx if i adds the transaction tx to its ledger (updating balances accordingly). The ledger is append-only, meaning that data can only be added (never deleted). That is, $\mathsf{Log}_i^t \subseteq \mathsf{Log}_i^{t'}$ for any $i \in N$ and $t' > t$.

Each node $i \in N$ can observe only its local history, which we call the node's *view*. In particular, a node may not know what other nodes observe. Depending on its view, the node may wait to receive messages, send messages, run computations, or finalize transactions (appending them to the ledger). In particular, a node can hold received transactions as "pending" before finalizing them.

A distributed protocol specifies each node's actions given its view. Nodes are said to be *honest* if they follow the protocol and are said to be *faulty* or *corrupt* otherwise. Correctness[13] of the consensus protocol is captured by the following[14] properties:

[10] The nodes aim to achieve agreement on the ledger in a sense that is related to game-theoretic common knowledge but is not equivalent to it. See Gonczarowski and Moses (2023) for a discussion of the differences and a relevant variation of common knowledge.

[11] Each transaction is cryptographically signed, ensuring that it is authorized by its owner. The analysis allows for a ledger that holds arbitrary data records.

[12] A user may issue a payment twice by mistake, or a malicious user may create conflicting transactions. The ledger must include only one of the conflicting transactions. Alternatively, the ledger can be viewed as a strict ordering of all finalized transactions, where a transaction is deemed invalid (and should be ignored) if it conflicts with earlier transactions.

[13] These properties are agnostic about the content of the ledger. Some applications impose additional *validity* properties that ensure that the content of the ledger is valid (e.g., the ledger can include only transactions that are properly signed).

[14] These definitions are simplified to focus on the core elements. Protocols that rely on randomness (e.g.,

- **Consistency:** There are no disagreements between finalized ledgers of honest[15] nodes. That is, for any honest nodes i and j (where i and j may be the same or different) and times t, t', either $\mathsf{Log}_i^t \subseteq \mathsf{Log}_j^{t'}$ or $\mathsf{Log}_i^t \supseteq \mathsf{Log}_j^{t'}$.

- **Liveness:** There exists a confirmation time T such that if an honest node i receives transaction tx at time t, then any honest node j finalizes tx by time $t + T$.

Informally, consistency ensures agreement among all the nodes that maintain local copies of the ledger, and liveness ensures that the ledger keeps updating. When both[16] of these properties hold, it appears to protocol users as if a single record keeper maintains a continuously updated ledger.

2.2 Free entry and permissionless protocols

Our interest is in protocols that provide an economically valuable property: nodes are exchangeable and may freely enter and exit. The following definition enables free entry.[17]

Definition 1. *We say the protocol is* ***permissionless*** *if a node can join or leave the protocol at any time.*

In particular, nodes do not require permission from any authority, centralized or distributed. Bitcoin's Nakamoto protocol (Nakamoto, 2008) is permissionless, allowing new nodes to join at any time.[18] By contrast, classic consensus protocols (e.g., Pease et al. 1980, Dolev and Strong 1983, Castro et al. 1999) are permissioned and assume that the set of nodes is known and fixed.

hash functions) may inevitably fail with some positive probability. The technical definition addresses this issue by requiring the probability of failure to be appropriately small; see Appendix B for details. In addition, we implicitly assume that the protocol has sufficient capacity to include all transactions.

[15] That is, the protocol guarantees agreement among parties who wish to agree on a common ledger. Although a deviating node can always record arbitrary content in its local ledger, consistency ensures that honest nodes remain unaffected by such deviations and do not accept an incorrect version of the ledger.

[16] While a consensus protocol is useful only if it satisfies both properties, it is useful to separate the two properties to distinguish different possible failures. Liveness is violated if the system stops processing records (e.g., a payment system that stops processing transactions because it has lost communication). Consistency is violated if two nodes disagree on a finalized record, then (e.g., a website gives a user confirmation that a payment has been processed although the payment failed).

[17] Running a node may require computational infrastructure, and node operators may need to pay the required costs to obtain it (Garratt and van Oordt, 2023, Prat and Walter, 2021). The definition requires no entry restrictions within the protocol.

[18] Moreover, nodes are anonymous to the Nakamoto protocol, and a newly joined node is treated identically to any other node.

When analyzing a run of the protocol from time t to time t', we say that a node is *online* if it joined before time t and remains active through time t'. We say a node is a *late joiner* if it joined at some time $\tau > t$. Note that consistency and liveness of the protocol must hold for both online nodes and late joiners.

2.3 Communication assumptions

We assume synchronous communication with dynamic joins and leaves (Pass and Shi, 2017b), unless explicitly stated otherwise. The same communication assumption is used in previous analyses of Nakamoto's protocol (Garay et al., 2015, Pass et al., 2017, Shi, 2022). We denote the maximum message delay by Δ. In other words, if an honest node i sends a message to an honest node j at time t, the message is guaranteed to be delivered at or before time[19] $t + \Delta$.

Nodes dynamically join and leave. If a node joins at some time t, then in each $t' \geq t$ it will have received all messages sent by honest nodes in round $t' - \Delta$ or earlier. Further, when a node joins, it may receive arbitrary messages from adversarial nodes. Note that late joiners cannot observe the timing of any messages sent before they joined.[20]

2.4 Definition of economic security

This section motivates and defines economic security for permissionless ledgers. Our goal is to provide users with meaningful guarantees, even if they are unaware of the protocol's technical specifics.

To motivate our definition, consider a merchant attempting to receive payment from a buyer. A dishonest buyer may want to deceive the merchant into believing that payment was made without transferring funds to the merchant. Fearing the buyer may be dishonest, the merchant might be reluctant to accept paper bills that might be counterfeit or a paper check that may not be backed by sufficient funds. To reduce the risk of fraud, the merchant may employ some security measures. For example, the merchant may check security devices on the bills or call the bank to verify that a check will be honored.

[19] The model allows for longer message delays if either the sender or the receiver is faulty.

[20] For example, suppose that an adversary attempts to broadcast a message at time t and falsely claims that this message was broadcast to all nodes at time $\tau < t - \Delta$. An online node can detect that the message was not broadcast at time τ because it did not receive it by time $\tau + \Delta$. However, a late-joining node cannot detect whether the message was sent at time t or earlier.

In a centralized digital payment system, a trusted recordkeeper (bank or payment processor) maintains a ledger of balances and transactions. A dishonest buyer may attempt to mislead the merchant into believing that payment was made when, in fact, the payment is not acknowledged by the trusted recordkeeper (and no funds are transferred to the merchant). To mitigate such risks, the merchant employs a point-of-sale terminal (computer) that verifies the payment. The merchant cannot be deceived if any transaction verified by the merchant's computer is recognized and approved by the trusted recordkeeper.

In an open distributed ledger, there is no trusted recordkeeper. Instead, a network of computer servers attempts to solve the canonical problem of distributed systems: collectively maintaining a single ledger. Each node holds a local copy of the ledger, and honest nodes update their local ledger according to the protocol's directions. If consistency holds, the many local copies held by different honest nodes agree. Consistency allows us to refer to *the* consensus ledger, although no canonical copy exists.[21]

Security for the merchant corresponds to the consistency of the protocol. Consider the merchant's point-of-sale terminal as a node in the distributed protocol. The merchant's node attempts to determine whether the transaction is included in the consensus ledger. When consistency holds,[22] a merchant's terminal that follows the protocol and determines that a transaction is finalized guarantees that the transaction is included in the consensus ledger (that is, it will be included in the local copy of the ledger of any honest node). In other words, the protocol's consistency guarantees the merchant cannot be deceived: any transaction finalized by the merchant must be accepted by any honest node.

Conversely, if an attacker deceives a merchant who follows the protocol, then consistency is violated.

We combine the distributed systems model with an economic model to define economic security. Classic results in the distributed systems literature prove the consistency and liveness of protocols given a bound on the number of faulty nodes. For example, classic consensus protocols such as Paxos (Lamport, 2001) and PBFT (Castro et al., 1999) guarantee consistency and liveness when more than 2/3 of the nodes are honest.[23] However, consistency

[21] One analogy is that one may refer to "the Bible" even though there is no canonical copy of the Biblical text.

[22] The consistency property is also appropriate for centralized payment systems, in which the merchant attempts to maintain consistency between the point-of-sale terminal and the trusted record keeper.

[23] These classic consensus protocols are permissioned, that is, designed for settings where a central designer authorizes each node (for example, a firm operating multiple data centers). In such settings, faults may occur due to hardware outages or software errors, and it is natural to assume that faulty behavior of many nodes is rare.

may be violated if a majority of nodes are faulty.

Permissionless protocols require a different approach. Under permissionless protocols, the number of faulty nodes is endogenous. An attacker can control many nodes that join the protocol or create financial incentives to convince existing node operators to deviate from the protocol. To give a meaningful quantification of the challenges the attacker must overcome in order to violate the protocol's consistency, we measure the attacker's cost of violating consistency. If the attacker cannot violate the protocol's consistency, we say the protocol has infinite economic security. If the attacker can violate consistency, quantifying the protocol's economic security requires an economic model specifying the attacker's possible actions and the cost of each possible action. Thus, our definition requires a specification of a distributed systems model and an economic model.

Definition 2. *Given a distributed systems model and an economic cost model, the economic security of a distributed ledger is the minimal attacker's cost required to violate consistency.*

A distributed ledger that guarantees a high level of economic security provides users with a trustworthy financial rail. Any user who wishes to observe the agreed-upon ledger can do so by operating an honest node and is assured that it will be difficult for an attacker to deceive them. We demonstrate in Appendix A that a distributed ledger that satisfies economic security and liveness is sufficient to construct a payment system. Note that our definition is stronger than certain equilibrium notions, as we allow the attacker to control a large coalition of deviating nodes.

Our definition takes a similar approach to Budish (2025) in that we focus on the attacker's cost, which can be quantified within the model. By contrast, the benefits to an attacker may depend on many aspects that cannot be observed within the model.[24] (Appendix D provides examples of the potential profitability of consistency violations.)

Knowledge of the protocol's economic security allows protocol users to adjust their behavior to ensure that an attacker cannot profitably deceive them (given reasonable assumptions on the attacker's benefit). For example, a merchant may only accept payments of a lower value than the protocol's economic security, making deception unprofitable for the attacker.

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

3 Incentives in Permissionless Protocols: Analysis of Bitcoin's Nakamoto Protocol

We formally demonstrate challenges in securing permissionless protocols by applying our definition of economic security to Bitcoin's Nakamoto protocol. Our analysis focuses on two distinct economic environments, chosen to highlight the inherent incentive deficiencies. First, the protocol lacks a mechanism to penalize attackers. Section 3.2 describes how the protocol can be attacked, and Section 3.3 details the design limitations that prevent it from detecting and punishing attacks. Second, even if an attack prompts a response from outside the protocol (e.g., an exchange rate collapse), this response negatively impacts all miners regardless of their involvement in the attack. Section 3.4 shows that if miners are small, the tragedy of the commons can arise.

These results suggest that Bitcoin's security does not stem from miners' monetary incentives. We provide an alternative explanation for the security of Bitcoin in Section 3.5. Other potential attack deterrents are discussed in Appendix E.

3.1 Background: Bitcoin's Nakamoto protocol

For completeness, we provide a brief description of the Nakamoto protocol. Some technical details are deferred to Appendix B.

The Blockchain data format The ledger is held in the format of a *blockchain*, which grows over time as new blocks are appended. A blockchain is an ordered sequence of blocks $(\mathsf{B}_0, \mathsf{B}_1, \ldots, \mathsf{B}_n)$, where B_0 is the protocol's genesis block. Each block B_ℓ contains transaction data and a reference to the preceding block $\mathsf{B}_{\ell-1}$. A block B_n uniquely identifies a chain of preceding blocks, and we write $\mathsf{Chain}(\mathsf{B_n}) = (\mathsf{B_0}, \mathsf{B_1}, \ldots, \mathsf{B_n})$. We say that block B_{n-k+1} is k-deep in $\mathsf{Chain}(\mathsf{B_n})$. Let $h(\mathsf{B}_\ell) = \ell$ denote the height of the block, defined as the number of blocks in $\mathsf{Chain}(\mathsf{B}_\ell)$ minus one.

The *view* of a node $i \in N$ encompasses all the information available to node i, including its local history and any blocks it has received. A node may receive multiple blocks of the same height, which cannot belong to the same chain.

Brief description of the Nakamoto protocol Each node (miner) organizes valid pending transactions into a suggested block. The protocol uses a hash function to select a node's

block randomly. Each node calculates the hash of its block,[25] and the block is valid if the hash is lower than the difficulty threshold D.[26] A node that finds a block with a valid hash is said to have mined a block. A node's mining power is the number of hash computations it can attempt per unit time. Nodes can write a special transaction into their block that awards them a block reward p_B.[27]

The protocol specifies the genesis block B_0 that all chains must extend. If a node mines or receives a block, it broadcasts the block to all other nodes. The protocol directs nodes to mine blocks that extend the *longest chain* in their view. Ties are broken arbitrarily.

The Nakamoto protocol itself does not specify when transactions are finalized. In common usage,[28] finality is parameterized by a required block depth k, where a transaction is considered finalized by node i if the transaction is within a block at least k-deep in the longest chain in i's view. That is, if $\mathsf{Chain}(\mathsf{B_n}) = (\mathsf{B}_0, \mathsf{B}_1, \ldots, \mathsf{B_n})$ is (one of) the longest chains in the view of node i, then i's local copy of the ledger is $\mathsf{Log}_i = (\mathsf{B}_0, \mathsf{B}_1, \ldots, \mathsf{B}_{n-k+1})$.

3.2 Economic security of Bitcoin under the rental model

We consider a stylized economic environment where miners can freely enter and exit. Several papers have explored a similar model and argued that an attacker can violate the consistency of Nakamoto at a low cost (Budish 2025, Tabarrok 2019, Auer 2019, Moroz et al. 2020, Gans and Halaburda 2024). We provide a similar result in our framework, showing that protocol users cannot defend against these attacks and clarifying why the protocol's design enables such attacks.

In the rental model, miners choose their computational power and may flexibly adjust it (for example, by renting capacity from a cloud computing service). Computing costs are linear: each miner pays c per attempt to compute a hash. Each hash provides the miner

[25] Following the literature (Garay et al., 2015, Pass et al., 2017, Shi, 2022), we model the hash function as a random oracle that always returns a random integer between $[0, 1]$ on a fresh input. We may assume that the execution proceeds in (possibly infinitesimally small) time increments, such that a unit of mining power can invoke the hash function once per increment. A miner can calculate a new hash for the same block by changing a nonce field. This field serves the sole purpose of allowing multiple inputs to the hash function for the same block.

[26] The difficulty threshold is periodically adjusted.

[27] A node that mines a block can include a special transaction that awards the node with a fixed number of newly minted coins and transaction fees from transactions included in its block. We collectively refer to these as the block reward and use p_b to denote its USD value. The node can claim the block reward only if the block is included in the consensus chain.

[28] The parameter k is commonly called the "confirmation depth". A common value for k is 6 blocks. For example, see https://coinmarketcap.com/academy/article/how-long-does-a-bitcoin-transaction-take.

a $1/D$ chance of mining a block. Each block includes a reward of p_{B}. Miners are purely profit-motivated. The attacker has the same capabilities and costs as other miners.

Consider an attacker who wishes to deceive a merchant into accepting an invalid transaction. That is, the merchant finalizes the transaction, but the transaction is not included in the consensus chain at some later time (e.g., when the merchant seeks to spend the funds). The merchant uses an observer node[29] to determine whether the transaction was finalized. Thus, to deceive the merchant, the attacker must violate the consistency of the protocol.

The following theorem states that Nakamoto's economic security in this economic environment is zero, as the attacker can violate consistency at zero cost. Moreover, this holds even if the merchant uses additional security measures, such as requiring additional verifications before finalizing transactions or monitoring network activity.

Proposition 1. *The economic security of Bitcoin's Nakamoto protocol in the rental model is zero; that is, the attacker can violate consistency and deceive a merchant at zero net cost. This holds even if the merchant perfectly monitors all network activity and requires additional confirmations for finality.*

The attacker deceives the merchant by creating a conflicting ledger: a blockchain in which the transaction crediting the merchant is absent. This is possible because of the permissionless nature of the protocol, as the attacker can construct such a blockchain by operating a set of miners and providing them with malicious inputs.[30] The attacker waits for the merchant to finalize the transaction before initiating the attack, rendering any monitoring by the merchant ineffective. The critical phase of the attack involves persuading honest nodes to adopt the conflicting ledger. This is achievable because of the longest chain rule: if the attacker's blockchain becomes longer than the honest chain, all honest nodes will adopt the conflicting chain, thereby violating consistency.

The proof uses a similar approach as Budish (2025), Auer (2019), and Gans and Halaburda (2024), and is included in Appendix C for completeness.[31] The key argument in the proof is that the protocol fails to recognize that an attack has occurred and treats blocks

[29] An observer node is identical to a mining node with zero hash power.

[30] The attacker's miners can even run the same code as honest miners. However, the attacker directs its miners to operate in a separate environment where they never receive the merchant's transaction or any blocks from a chain that includes it.

[31] Budish (2025) and Auer (2019) show that Nakamoto has positive economic security if the attacker cannot recoup the mining rewards or if the attacker has higher mining costs. Gans and Halaburda (2024) show that the net cost of the attack can be negative because the attacker can affect the block reward. We discuss such factors in Section E.

mined by the attacker as if they were mined by honest miners. In particular, once the attacker's chain is adopted, the attacker becomes the "honest miner" who created the agreed ledger and is paid the mining rewards. Since honest miners do not lose from mining, neither does the attacker.

Observe that the result holds regardless of the amount of mining rewards in the protocol. This is because mining rewards are given equally to honest and attacking miners. However, deterrence of attacks requires that an attacker's payoff is lower than an honest miner's payoff.

3.3 Why Nakamoto fails to detect attacks

To understand why the Nakamoto protocol permits such attacks, it is useful to examine its design and the rationale behind its longest-chain rule.

If all nodes are honest and messages are communicated instantly, exactly one block is mined at each height, and nodes achieve consensus on this unique 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.[32]

To address this possibility, the Nakamoto protocol uses the longest-chain rule to reach consensus on a unique chain. If conflicting blocks of the same height exist, the next mined block extends and selects one unique chain. Although there is a possibility that both blocks are extended simultaneously, the probability of continuing conflict becomes small when network lags are sufficiently short (relative to inter-block time). The probability that multiple honest miners mine conflicting blocks "simultaneously" is small, and the probability that two conflicting chains $(\mathsf{B}_0, \ldots, \mathsf{B}_{n-1}, \mathsf{B}_n, \ldots, \mathsf{B}_{n+k})$ and $(\mathsf{B}_0, \ldots, \mathsf{B}_{n-1}, \mathsf{B}'_n, \ldots, \mathsf{B}'_{n+k})$ are both mined by honest nodes is exponentially small in k. This observation is used to show that the Nakamoto protocol satisfies[33] consistency and liveness if (i) communication is synchronous: network lags are bounded and sufficiently short, and (ii) a sufficient fraction of the mining power is controlled by honest nodes.

Both assumptions are necessary: the Nakamoto protocol can lose consistency if network lags are too long or the adversary controls enough computational power. To see why synchrony is necessary, consider a network partition. That is, suppose that some miners are

[32]For example, assume that it takes 10 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 view of any node includes two conflicting blocks $\mathsf{B}_n, \mathsf{B}'_n$, and it is unclear which block should be considered part of the consensus chain.

[33]See Appendix B and Garay et al. (2015), Pass et al. (2017), Shi (2022), and Guo and Ren (2022).

in North America (NA) and some are in 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 view.[34] Two conflicting 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 this point, the protocol reconciles the two conflicting chains by directing all nodes to adopt the longest chain. If the partition lasts long enough, this reconciliation can violate consistency: NA and SA finalize conflicting blocks that are k-deep, and the group with the shorter chain discards some finalized blocks.

An attacker with sufficient computational power essentially exploits the longest-chain rule by "pretending" to be partitioned. If the attacker can mine a conflicting chain that is longer than the honest chain, the protocol directs all miners to adopt the new longest chain, as in the reconciliation.

In practice, the community will be aware that the protocol was attacked. Under synchronous communication, which is necessary for Nakamoto (Pass and Shi, 2017a), honest nodes can detect that the attacker's blocks could not have been mined by honest nodes because they receive these blocks after a long delay. The protocol cannot detect whether this delay results from the attacker's malicious behavior or from honest nodes suffering a network partition. However, it is very unlikely that a significant fraction of the computing power in the network is partitioned.[35] A common assumption in the literature is 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 can detect attacks.

3.4 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 obtain mining power only from existing miners. Even though the Nakamoto protocol is permissionless, external factors may restrict miner entry. For example, miners may employ specialized mining equipment or have access to other restricted resources (e.g., cheap electricity and computing facilities). This stylized economic environment can be

[34] Partitioned nodes in SA can observe that they do not receive any messages from NA nodes but cannot determine whether NA nodes have stopped running the protocol or whether there is a communication failure. The protocol directs SA nodes to continue operating as if all NA nodes stopped.

[35] Moreover, if such a significant partition occurs, it will likely be known to the nodes and protocol users.

considered as an extreme fixed cost[36] of mining, which makes entry prohibitively expensive.

To focus on the role of economic incentives, we assume that miners are rational and profit-driven. The attacker can affect the behavior of miners by committing to an incentive contract, which we refer to as a bribery contract (Bonneau, 2016, McCorry et al., 2018, Winzer et al., 2019, Judmayer et al., 2021). We refer to this economic environment as the *bribery model*.

The bribery model allows miners to suffer losses due to a community response to an attack. When there is no free entry of miners, it is possible that $p_{\mathsf{B}} > D \cdot c$ and miners earn positive profits from mining, and the expectation of future profits increases the value of mining equipment. If the community responds to an attack by decreasing usage, the value of future mining rewards decreases, and mining equipment loses value. Miners may also incur additional losses, and we use Ψ_A to capture all losses miners suffer if the system is attacked.[37]

A community response may carry additional implications for the attacker. We consider these separately below and in Section 3.5. In the bribery model, a successful attack allows the attacker to collect the standard value of block rewards p_{B} despite the community response.

The bribery model has a continuum of miners, each controlling an infinitesimal unit of computational power.[38] Let ψ_A denote the normalized harm per infinitesimal miner. Miners will participate in the attack if it is in their selfish best interest, given the payments they stand to receive from the protocol, the bribes from the miner, and the harm they may suffer from an attack.

A bribery contract induces a game among the miners. For simplicity, we assume miners play a simultaneous move game where all miners observe the bribery contract and simultaneously choose whether to participate in the attack. The attacker succeeds in attacking the protocol if the attacker violates the protocol's consistency under any equilibrium of the induced game.

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. *In the bribery model, the economic security of Bitcoin's Nakamoto protocol is arbitrarily close to zero; that is, the attacker can violate consistency and deceive a merchant*

[36] Garratt and van Oordt (2023) analyze a security model with fixed costs and Prat and Walter (2021) analyze miner entry with fixed costs.

[37] Garratt and van Oordt (2023) endogenously determine the miners' loss and evaluate the security implications for an attacker who owns all the mining equipment used in the attack.

[38] Contemporaneously and independently of this work, Newman (2023) analyzes a model with finitely many miners and shows that small miners are unlikely to be pivotal to an attack's success.

at a net cost arbitrarily close to zero.

Prof of Proposition 2. As in the proof of Proposition 1 in Appendix B, the attacker waits until the merchant finalizes the transaction in some block B_n at some time t. Only then, the attacker publishes a bribery contract that pays miners who mine blocks $\mathsf{B}'_n, \ldots, \mathsf{B}'_{n+k-1}$ that form the attack chain $(\mathsf{B}_0, \ldots, \mathsf{B}_{n-1}, \mathsf{B}'_n, \ldots, \mathsf{B}'_{n+k-1})$. The contract commits to pay $\tilde{p}_b$ for each mined block.[39] The block reward from each of the blocks $\mathsf{B}'_n, \ldots, \mathsf{B}'_{n+k-1}$ is directed to an address controlled by the attacker, enabling the attacker to collect them if the attack succeeds.

	Attack Succeeds	Attack Fails
Mine attack chain	$x \cdot (\tilde{p}_\mathsf{B}/D - c) - \psi_A$	$x \cdot (\tilde{p}_\mathsf{B}/D - c)$
Mine honest chain	$-x \cdot c - \psi_A$	$x \cdot (p_\mathsf{B}/D - c)$
Do not mine	$-\psi_A$	0

Table 1: Miner payoffs under the bribery contract.

This bribery contract induces a game among miners. Each miner may choose to mine the honest chain, mine the attack chain, or stop mining.[40] 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. The bribery contract guarantees to pay miners who mine a block in the attack chain an attack block reward of $\tilde{p}_\mathsf{B}$ regardless of whether the attack succeeds or fails. Miners who mine a block in the honest chain collect an honest block reward of p_B only if the attack fails. If the attack succeeds, all miners suffer the cost of the attack ψ_A (recall that miners are anonymous). Let x denote the number of hashes a miner can compute during the attack. The potential payoffs of a miner are summarized in Table 1.

By assumption, miners are small and not pivotal to the attack's success. Because honest miners must be willing to mine the honest chain when there is no attack, we have that $p_\mathsf{B}/D - c \geq 0$. Thus, if the attacker pays $\tilde{p}_\mathsf{B} > p_\mathsf{B}$, 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 $\tilde{p}_\mathsf{B} - p_\mathsf{B} \approx 0$, and the net cost of the entire attack is arbitrarily close to zero. □

[39] Such a commitment can be implemented by a smart contract running on a different blockchain (McCorry et al., 2018). To receive payment, the miner submits the block to the smart contract, which can verify that the block extends the attack chain.

[40] For simplicity, we assume miners do not split their mining power.

Notably, the attacker can induce miners to participate in the attack without compensating miners for the total harm of the attack, Ψ_A, or even for their individual losses, ψ_A. The game induced by the bribery contract creates a tragedy of the commons for miners. Even if Ψ_A is substantial, it does not affect an individual miner's incentive to deter the attack. Miners are anonymous, and the attack harms both miners who participate in the attack and honest miners alike. If an individual miner cannot affect whether the attack succeeds or fails, the harm does not incentivize individual miners to deter attacks. In that respect, Proposition 2 gives a foundation for the EAAC (expensive to attack in the absence of collapse) property of Budish et al. (2024).

The community response may limit the attacker's ability to recoup its costs by collecting mining rewards. For example, suppose that the community responds to an attack by refusing to accept payments through the protocol. In that case, the attacker cannot collect block rewards after the attack. This response increases the attacker's net cost to p_{B} per block in the attack chain, but the attacker does not need to compensate miners for the losses from the attack, Ψ_A. Such a response differentiates between honest miners and attackers, creating penalties that deter attacks. However, other community responses can be more effective. We discuss these next.

3.5 What deters attacks in practice?

Despite the possibility of attacks, major cryptocurrencies like Bitcoin and Ethereum[41] have remained secure. At the same time, numerous attacks on multiple cryptocurrencies that operate the Nakamoto protocol show that the Nakamoto protocol is indeed vulnerable to attacks in practice (Shanaev et al., 2019).

We discuss several explanations for the absence of attacks on Bitcoin in Appendix E and highlight one explanation in particular: the community's ability to correct the ledger after an attack, which we refer to as a *community response*. Recall that the protocol provides a record-keeping service: maintaining an agreed-upon ledger. If the community agrees that the ledger was corrupted, the community may override the protocol to change it. In blockchain terminology, the community can "fork" the protocol by directing all miners to ignore the attack chain and extend the honest chain. Doing so reverts any effect of the attack, leaving the ledger unchanged by the attack.[42] Moreover, the community response imposes a significant

[41] Ethereum used the Nakamoto protocol until 2022.

[42] Such corrections also occur in traditional record-keeping services. For example, the New York Stock Exchange canceled trades that resulted from a software glitch (CNBC, 2024).

risk to the attacker of having to pay all the attack's costs only to see it undone.

Such community responses are not without precedent and are actively considered by system developers.[43] A notable example is the Ethereum community's decision to fork the Ethereum ledger in order to return funds stolen from the DAO attack (Ethereum Foundation, 2016). Such measures, however, require the community to be willing to override the protocol and able to agree on the correct ledger. While the Ethereum blockchain community was able to override the protocol, the Ethereum Classic blockchain rejects any community interventions (Ethereum Classic, 2022). Anecdotally, Ethereum did not suffer any consistency attacks, whereas Ethereum Classic suffered multiple such attacks (Nesbitt, 2019).

However, the community response is not a silver bullet. To undo the attack, the community must reach an agreement on the correct ledger. In other words, the community needs to solve the same consensus problem that the protocol attempts to solve. Moreover, the attacker may employ a different attack to deceive both the protocol and the community. Without a formal model of the community's decision-making process, it is difficult to assess the community response.

In the next section, we analyze a model inspired by the efficacy of the community response. The protocol can be seen as incorporating the community response within the protocol. The analysis evaluates the required capabilities and limitations of the community.

4 Possibility of a Permissionless and Economically Secure Consensus Protocol

We now consider the broader question of protocol design: Can a permissionless[44] protocol achieve economic security? This section answers affirmatively. We show that a permissionless protocol can be economically secure by constructing such a protocol. This protocol is a variation of the Nakamoto protocol, and we name it *Stubborn Nakamoto* protocol. We prove that it is infinitely economically secure – that is, even an attacker with unlimited resources

[43] For example, Buterim writes "if the cryptoeconomic consensus fails, whether due to a bug or an intentional 51% attack, a vast community of many thousands of developers and many more users are watching carefully to make sure the chain recovers correctly" (Buterin, 2023). The developer community often uses the terms social consensus (Buterin, 2016) or a community hard fork when referring to the protocol's reliance on external sources.

[44] A common alternative approach is to modify the protocol's permissionless properties. For example, Proof of Stake (PoS) protocols require nodes to register an identifier and associate it with an account. We discuss such protocols in Section 5.

cannot violate consistency. The Stubborn Nakamoto protocol preserves the permissionless properties of the Nakamoto protocol while ensuring economic security. Specifically, miners remain anonymous and can enter or exit the system without restriction.

The design approach underlying the Stubborn Nakamoto protocol is motivated by the analysis in Section 3. The negative results in Section 3 demonstrate that the inability to detect attacks compromises the economic security of Bitcoin's Nakamoto protocol, whereas the effectiveness of the community response highlights the power of attack detection. Accordingly, the Stubborn Nakamoto protocol is designed to enable nodes to detect potential attacks.

The protocol can be interpreted as incorporating the community response which restores the correct ledger following an attack. A straightforward design might require nodes to communicate and reestablish the correct ledger after an attack. The Stubborn Nakamoto protocol takes a more efficient approach: if a node is assured that the community response will restore the correct ledger, it can immediately adjust its local copy to reflect that correct ledger. (For example, a node can reject an attack chain if it is confident that all honest nodes will likewise reject it.) The Stubborn Nakamoto protocol provides nodes with sufficient knowledge about the behavior of other nodes, ensuring that they maintain a consistent ledger.

This analysis in this section also examines the role of the "community" – that is, external inputs beyond the protocol. Some external input is unavoidable. For instance, the initialization of any protocol requires an external source (e.g., setting the genesis block in Bitcoin's Nakamoto protocol), and late joiners must rely on an external input to connect to the protocol (e.g., distinguishing Bitcon's chain from Dogecoin's chain). However, reliance on such external inputs should be minimized.[45] This section rigorously evaluates the necessary role of the community, formalizing it as a *recovery oracle*. We prove that some external dependence is necessary for recovery after attacks, and design the Stubborn Nakamoto protocol to minimize this reliance in two key ways. First, the requirements from the recovery oracle can be practically implemented. Second, the Recovery Oracle should not hold excessive control over the system. The Stubborn Nakamoto protocol is thus designed to minimize both: the recovery oracle is simple to implement, invoked only for attack recovery, and holds minimal influence over the system.

[45]Reliance on the community to resolve challenges is often challenging to formally model and analyze. Moreover, reliance on out-of-protocol inputs can introduce fundamental issues. First, the primary objective of the protocol is to establish consensus, so it should not depend on the community's ability to achieve consensus (conversely, if the protocol cannot establish consensus, why should the community be able to do so?). Second, reliance on external inputs may undermine the protocol's permissionless properties, potentially giving control over the system to an out-of-protocol entity.

Section 4.1 defines the Stubborn Nakamoto protocol, and we prove its security and liveness for online nodes in Section 4.2. Section 4.3 defines the protocol for late joiners and proves the corresponding guarantees. Section 4.4 details how the Stubborn Nakamoto protocol can recover from an attack and formalizes the role of the community in the recovery. Section 4.5 examines the necessary role of the community.

4.1 The Stubborn Nakamoto protocol

In contrast to the analysis of Bitcoin's Nakamoto protocol in Section 3, the definition and analysis of Stubborn Nakamoto protocol require careful consideration of each node's knowledge, and in particular each node's knowledge of the state of other nodes. Before giving the definition of the Stubborn Nakamoto protocol, we introduce notation and definitions that represent the knowledge held by a node. Note that this section defines the Stubborn Nakamoto protocol for online nodes. The protocol for late joiners is defined in Section 4.3.

The Stubborn Nakamoto protocol makes the same communication assumptions as Bitcoin's Nakamoto protocol, namely synchronous communication with late joiners. We also assume "implicit echoing", namely, if an honest node[46] receives a message at time t, then every honest node receives the message by time $t + \Delta$. Stubborn Nakamoto also shares the same setup assumptions as Bitcoin's Nakamoto protocol: all nodes are assumed to have knowledge of the genesis block and the protocol's rules. The protocol is parameterized by a confirmation depth k.

Stubborn Nakamoto uses the blockchain data format. The local view of node i includes all the blocks received by i, as well as additional information recorded in the form of block labels. We write $\mathsf{B} \in \mathsf{View}_i^t$ to denote that node i received block B by time t. Each block label is encoded as a subset of blocks in view. We describe below how a node assigns each label given its local view. Labels are never removed.[47]

A block B *conflicts* with block B', denoted $\mathsf{B} \not\sim \mathsf{B}'$, if there is no chain that contains both B, B'. That is, neither $\mathsf{Chain}(\mathsf{B}) \subseteq \mathsf{Chain}(\mathsf{B}')$ nor $\mathsf{Chain}(\mathsf{B}') \subseteq \mathsf{Chain}(\mathsf{B})$. Given a block B and a set of blocks $\mathcal{A}$, if $\nexists \mathsf{B}' \in \mathcal{A}$ such that $\mathsf{B} \not\sim \mathsf{B}'$ we say that B does not conflict with $\mathcal{A}$, written $\mathsf{B} \sim \mathcal{A}$. We write $\mathsf{B} \not\sim \mathcal{A}$ if $\exists \mathsf{B}' \in \mathcal{A}$ such that $\mathsf{B} \not\sim \mathsf{B}'$.

The protocol may instruct a node to *halt* if the node detects a potential conflict between its local view and the local view of another honest node. When a node halts it stops its

[46] We assume that the implicit echoing assumption holds for all nodes, including observer nodes that do not mine. Sridhar et al. (2024) show the necessity of this assumption.

[47] For example, the set $\mathcal{C}_i^t$, defined in Definition 3, satisfies $\mathcal{C}_i^t \subseteq \mathcal{C}_i^\tau$ for $\tau \geq t$ (and analogously for all other block labels defined below).

attempts to mine any new blocks and stops certifying any new blocks (as described below). If the node has already certified some blocks, it completes the steps below to determine whether these blocks should be finalized.

The Stubborn Nakamoto protocol relies on the following definition, which specifies the first step a node takes toward finalizing a block. The conditions for this first step closely resemble the conditions for finalizing a block in the Nakamoto protocol. However, unlike the Nakamoto protocol, the required confirmation depth k is explicitly defined within the protocol, and all nodes use the same confirmation depth k.

Definition 3. *Node i **certifies** block B at time t, written $\mathsf{B} \in \mathcal{C}_i^t$, if at time t there is a chain $(\mathsf{B}_0, \ldots, \mathsf{B}, \ldots, \bar{\mathsf{B}})$ in View_t^i such that*

1. *Block B is at least k-deep in the chain $\mathsf{Chain}(\bar{\mathsf{B}})$, i.e., $h(\bar{\mathsf{B}}) - h(\mathsf{B}) + 1 \geq k$.*
2. *The chain $\mathsf{Chain}(\bar{\mathsf{B}})$ is a longest chain in view, meaning $h(\bar{\mathsf{B}}) \geq h(\tilde{\mathsf{B}})$ for any $\tilde{\mathsf{B}} \in \mathsf{View}_i^t$.*

*Additionally, if in the view of node i at time t there is no certified block B' that conflicts with B, then B is also **clear-certified**, denoted $\mathsf{B} \in \mathcal{CC}_i^t$.*[48]

The protocol relies on the following definition to characterize each node's knowledge of what other nodes may have certified.

Definition 4. *For $\gamma \geq 0$, a block B becomes **γ-potentially certified** by node i at time t, written $\mathsf{B} \in \mathcal{PC}_i^t(\gamma)$, if at time t there is a chain $(\mathsf{B}_0, \ldots, \mathsf{B}, \ldots, \bar{\mathsf{B}})$ in View_t^i such that*

1. *Block B is at least k-deep in the chain $\mathsf{Chain}(\bar{\mathsf{B}})$, i.e., $h(\bar{\mathsf{B}}) - h(\mathsf{B}) + 1 \geq k$;*
2. *The chain $\mathsf{Chain}(\bar{\mathsf{B}})$ would have been a longest chain in i's view at time $t - \gamma$, meaning $h(\bar{\mathsf{B}}) \geq h(\tilde{\mathsf{B}})$ for any $\tilde{\mathsf{B}} \in \mathsf{View}_i^{t-\gamma}$.*

We can now state the definition of the Stubborn Nakamoto. Recall that each node i maintains Log_i^t, a local copy of the ledger that consists of all finalized blocks. The local copy of the ledger Log_i^t is required to agree with the local copy of the ledger Log_j^τ of any other honest node j.

The Stubborn Nakamoto protocol directs nodes to take two steps before finalizing a block and adding it to their local copy of the ledger. First, the block needs to be certified. Second, the node waits to see that no other node could have certified a conflicting block. If the node does not observe any potential conflicts, the node can finalize the block and add it to the ledger.

[48] If two conflicting blocks both become certified at time t, neither is clear-certified.

Definition 5 (The Stubborn Nakamoto Protocol)**.** *As in Nakamoto, the protocol directs nodes (miners) to attempt to mine a block that extends the longest chain in view. Each node i takes the following steps*[49] *towards finalizing blocks:*

1. *Node i certifies blocks according to definition 3. If a block B is certified but is not clear-certified (that is, in the local view of node i there is a block B' that is certified and conflicts with B), then node i halts.*

2. *If i certified block B at time t_1, then at time $t_2 = t_1 + 2\Delta$ node i checks if there is a block B' in i's local view that conflicts with B and is 2Δ-potentially certified. If yes, node i halts. If B does not conflict with any 2Δ-potentially certified block in i's local view, then node i finalizes B, written $\mathsf{B} \in \mathsf{Log}_i^{t_2}$.*

3. *If i certified block B at time t_1, then at time $t_3 = t_1 + 4\Delta$ node i checks if there is a block B' in i's local view that conflicts with B and is 4Δ-potentially certified. If yes, node i halts.*

The definition of the Stubborn Nakamoto protocol is conceptually simple yet carefully designed. On one hand, the protocol ensures that nodes can detect and prevent conflicts with other nodes. On the other hand, nodes must not be overly cautious, as halting too easily would hinder progress and compromise the protocol's liveness. The protocol's design, along with the definition of γ-potentially confirmed blocks, carefully balances these competing requirements.

One interpretation of the protocol is that it incorporates the community response (discussed in Section 3.5) into the protocol itself. Because the Stubborn Nakamoto protocol is permissionless, it cannot prevent an attacker with sufficient resources from creating a conflicting ledger (as described in Section 3.2). Despite this, the protocol prevents consistency violations by allowing nodes to identify attacks. If a node can identify received blocks as attack blocks that could only have been mined by an attacker, it can simply ignore them. However, to maintain agreement, we must ensure that if node i ignores a block, it can be confident that any honest node j will also ignore that block.

[49] Using notation: Let t_1 be the earliest time such that $\mathsf{B} \in \mathcal{C}_i^{t_1}$.

1. If $\mathsf{B} \not\sim \mathcal{C}_i^{t_1}$, then node i halts at time t_1.
2. Let $t_2 = t_1 + 2\Delta$. If $\mathsf{B} \not\sim \mathcal{PC}_i^{t_2}(2\Delta)$, then node i halts at time t_2. If $\mathsf{B} \sim \mathcal{PC}_i^{t_2}(2\Delta)$, then $\mathsf{B} \in \mathsf{Log}_i^{t_2}$.
3. Let $t_3 = t_1 + 4\Delta$. If $\mathsf{B} \not\sim \mathcal{PC}_i^{t_3}(4\Delta)$, then node i halts at time t_3.

To ensure agreement among all honest nodes, the Stubborn Nakamoto protocol differs from the standard Nakamoto in two key ways. First, it specifies an explicit condition for finality, ensuring that an honest node can reliably predict the behavior of other honest nodes. Second, before finalizing a block, nodes wait for a conflict-discovery period, allowing them to learn if another node receives a conflicting ledger. While an attacker with unlimited resources may still create a conflicting ledger, honest nodes will be able to detect the existence of multiple potential ledgers, identify the risk of disagreement, and halt if necessary. More generally, the protocol guarantees that an honest node finalizes a block only if that node is assured that no other honest node could have finalized a conflicting block.

The Stubborn Nakamoto protocol prioritizes consistency and instructs nodes to finalize a block only if consistency is assured. Given the known impossibility of consensus without an honest majority (Pass and Shi, 2017a, Lewis-Pye and Roughgarden, 2023, Sridhar et al., 2024), any protocol must, under some attacks, sacrifice either consistency or liveness. The protocol may lose liveness under some attacks that cause nodes to detect a conflict and halt. We discuss how the protocol can regain liveness after an attack in Section 4.4.

Remark 1. *The Stubborn Nakamoto protocol can be made more resilient by instructing nodes to become "stubborn" on any block for which they completed step 3, meaning a node ignores any block that conflicts with a block that completed step 3 without halting. This modification allows online nodes to completely ignore the attack described in Section 3.2, and it does not affect the protocol's security and liveness guarantees for online nodes which are proved in Section 4.2. However, such a modification will require careful treatment of late joiners, which we discuss in Section 4.3 and Section 4.4.*

4.2 Consistency and liveness of Stubborn Nakamoto

We turn to prove that Stubborn Nakamoto guarantees consistency and liveness for online nodes. Treatment of late joiners is in Section 4.3. Before presenting our main theorem, we first establish formal results that characterize the knowledge an honest online node has about other honest online nodes under the Stubborn Nakamoto protocol.

Lemma 1. *If an honest node i certifies block B at time t, then by time $t + \Delta$ block B is 2Δ-potentially certified by any honest node j.*
If an honest node i γ-potentially certifies block B at time t, then by time $t + \Delta$ block B is $(\gamma + 2\Delta)$-potentially certified by any honest node j.

Proof. Because $\mathcal{C}_i^t = \mathcal{PC}_i^t(0)$, it is sufficient to prove the second part of the lemma. Assume that block B is γ-potentially certified by honest node i at time t. That is, there is a chain $\mathsf{Chain}(\bar{\mathsf{B}}) = (\mathsf{B}_0, \ldots, \mathsf{B}, \ldots, \bar{\mathsf{B}}) \subseteq \mathsf{View}_i^t$ that satisfies the conditions of definition 4 given the local view of node i at time t, namely: $h(\bar{\mathsf{B}}) - h(\mathsf{B}) + 1 \geq k$, and $h(\bar{\mathsf{B}}) \geq h(\tilde{\mathsf{B}})$ for any $\tilde{\mathsf{B}} \in \mathsf{View}_i^{t-\gamma}$.

Consider arbitrary honest node j. Because node i receives all the blocks in $\mathsf{Chain}(\bar{\mathsf{B}})$ by time t, node j receives all blocks in $\mathsf{Chain}(\bar{\mathsf{B}})$ by time $\tau \leq t + \Delta$. We have that B is k-deep in $\mathsf{Chain}(\bar{\mathsf{B}})$, satisfying $h(\bar{\mathsf{B}}) - h(\mathsf{B}) + 1 \geq k$. To see that $h(\bar{\mathsf{B}}) \geq h(\tilde{\mathsf{B}})$ for any $\tilde{\mathsf{B}} \in \mathsf{View}_j^{\tau-\gamma-2\Delta}$, observe that

$$\mathsf{View}_j^{\tau-\gamma-2\Delta} \subseteq \mathsf{View}_j^{t+\Delta-\gamma-2\Delta} = \mathsf{View}_j^{t-\gamma-\Delta} \subseteq \mathsf{View}_i^{t-\gamma}$$

where the first inclusion follows because $\tau \leq t + \Delta$, and the last inclusion follows from communication between two honest nodes. Thus, we have that $\mathsf{B} \in \mathcal{PC}_j^{\tau}(2\Delta + \gamma) \subseteq \mathcal{PC}_j^{t+\Delta}(2\Delta + \gamma)$. □

We can now state our main result, showing that Stubborn Nakamoto is economically secure.

Theorem 1 (Consistency of Stubborn Nakamoto)**.** *Under synchrony, Stubborn Nakamoto satisfies consistency even in the presence of an adversary with unbounded resources*[50] *and any number of corrupted nodes.*

Proof. Suppose that honest node i finalized block B at time t_2. Let $t_1 = t_2 - 2\Delta$ denote the time when node i certified block B. We need to show that if some honest node j finalized a block B' then it must be that $\mathsf{B}' \sim \mathsf{B}$. It is immediate from definition 3 that node i never clear-certifies a block $\mathsf{B}' \not\sim \mathsf{B}$, and therefore node i cannot finalize a block B' that conflicts with B.

For the sake of contradiction, consider an honest node $j \neq i$ that finalized a block $\mathsf{B}' \not\sim \mathsf{B}$ at time t_2', and let $t_1' = t_2' - 2\Delta$ denote the time node j certified block B'. By Lemma 1, node j must have 2Δ-potentially certified block B by time $t_1 + \Delta$. Because node j finalized block $\mathsf{B}' \not\sim \mathsf{B}$, we must have that $t_1' + 2\Delta = t_2' < t_1 + \Delta$. Symmetrically, by Lemma 1 we have that node i must have 2Δ-potentially certified block B' by time $t_1' + \Delta$. Because node i finalized block $\mathsf{B} \not\sim \mathsf{B}'$, we must have that $t_1 + 2\Delta = t_2 < t_1' + \Delta$. But this implies that $t_1 < t_1' - \Delta$ and $t_1' < t_1 - \Delta$, a contradiction.

[50] That is, the attacker can control arbitrarily mining resources with an unbounded hash rate. We maintain the standard assumption that the attacker cannot break cryptographic primitives.

□

The consistency guarantee provided by Theorem 1 is notable for two key reasons. First, it demonstrates that the Stubborn Nakamoto protocol is maximally economically secure, as even an attacker with unlimited resources cannot violate consistency and deceive the protocol's users. Second, the protocol's security does not rely on mining rewards or other monetary incentives.[51]

Of course, a protocol is only useful if it also satisfies liveness. To see why Stubborn Nakamoto also satisfies liveness, observe that Stubborn Nakamoto differs from Nakamoto only when, in the local view of some honest node i, there are two conflicting blocks $\mathsf{B}' \not\sim \mathsf{B}$ that are both γ-potentially certified for $\gamma \leq 4\Delta$. In other words, nodes will act differently under the two protocols only in realizations that are close to the Nakamoto protocol's consistency violations. Under approximately the same conditions that guarantee consistency and liveness of the Nakamoto protocol, such realizations do not arise, and Stubborn Nakamoto satisfies consistency and liveness.

Theorem 2 builds on these ideas to formally establish that Stubborn Nakamoto satisfies consistency and liveness under a sufficient honest majority assumption, the same assumption required for Nakamoto to satisfy consistency and liveness. The exact technical statement of the network-aware honest majority assumption and the proof can be found in Appendix B.

Theorem 2 (Liveness of Stubborn Nakamoto)**.** *Under synchrony, Stubborn Nakamoto satisfies consistency and liveness if the network-aware honest majority assumption holds.*

Roughly speaking, both Nakamoto and Stubborn Nakamoto require the same honest majority assumption to guarantee both consistency and liveness. The protocols differ when this assumption is not satisfied.[52] Under the Nakamoto protocol, an attack that controls sufficiently many nodes can cause consistency violations. In contrast, the Stubborn Nakamoto protocol prevents consistency violations by allowing nodes to detect when consistency might be at risk. Rather than allowing transactions to be added to the ledger only to be later reverted, Stubborn Nakamoto instructs nodes to halt and wait for a safe recovery. Effectively, the Stubborn Nakamoto protocol transforms Nakamoto's consistency violations into liveness

[51] Mining rewards play a role in deterring attackers from compromising the protocol's liveness; see Section 4.4.

[52] As we discuss below, some trade-offs in protocol design are necessary, as no protocol can guarantee both consistency and liveness under corrupt majority (Pass and Shi, 2017a, Lewis-Pye and Roughgarden, 2023, Sridhar et al., 2024).

violations. This behavior is particularly desirable in financial systems, where transaction integrity is paramount.

Note that the version of Stubborn Nakamoto outlined in Remark 1 can completely avoid some of Nakamoto's consistency violations without suffering a liveness violation. In particular, this version of the protocol maintains both consistency and liveness under the attack outlined in Section 3.2.

The proof of Theorem 2 uses the intermediate stochastic bounds used to prove the consistency and liveness of Nakamoto. While the scenarios under which liveness of Stubborn Nakamoto is violated are very close to those under which consistency of Nakamoto is violated, it is possible to have a liveness violation of Stubborn Nakamoto without a corresponding consistency violation of Nakamoto. For this reason, the proof of Theorem 2 requires subtle changes to the analysis of Nakamoto consensus. We state the technical details, required lemmas, and the proof of Theorem 2 in Appendix B.

4.3 Late joiners

This section establishes guarantees for late joiners under Stubborn Nakamoto. A late joiner is a node that joins the system at some time t after the protocol has already been running for some time, and some blocks may already have been finalized. Recall that we adopt the standard communication assumption that by time $t+\Delta$, the late joiner receives all messages sent by honest nodes by time t or earlier (Pass and Shi, 2017b).[53] However, a late joiner node cannot distinguish the time at which earlier messages were sent.[54] Additionally, a late joiner may also receive arbitrarily many messages from adversarial nodes.

Allowing late joiners is crucial for the free entry of nodes. Without this capability, all nodes would be required to join at the protocol's inception and remain continuously online. However, ensuring consistency for late joiners presents significant challenges. For example, the well-known Dolev-Strong protocol guarantees consistency and liveness for online nodes but fails to provide these guarantees for late joiners (see, e.g., Pass and Shi (2017a)).

[53] The same communication assumption is also used by the standard Nakamoto protocol, which requires late-joining nodes to receive and verify the entire chain of previous blocks, linking back to the genesis block. The delivery of past messages is essential in any protocol to ensure that newly joining nodes can correctly establish and continue the ledger.

[54] That is, if $\tau_1, \tau_2 < t$, a late joiner cannot tell whether a message was sent at time τ_1 or at time τ_2. This is in contrast to online nodes, which are able to observe when any message is sent by an honest node up to a Δ time lag. In particular, late joiners cannot recreate the finalization steps taken by online nodes under Stubborn Nakamoto, as they cannot determine which blocks were received within 4Δ of a block's certification.

The following definition formalizes the process by which a late joiner node becomes an online node. We refer to these steps as the node's wake-up procedure.

Definition 6 (Late Joiner Wake-Up)**.** *Let honest node i be a late joiner that awakens at time t_w. Node i waits until time $t_{on} > t_w$ such that node i does not receive any new blocks between $t_{on} - 6\Delta$ and t_{on}. If there are two conflicting blocks $\mathsf{B}, \mathsf{B}' \in \mathsf{View}_i^{t_{on}}$ that are both k-deep in some longest chain in $\mathsf{View}_i^{t_{on}}$, then node i halts. Otherwise, node i certifies and finalizes all blocks that are k-deep in a longest chain in view, and becomes an online honest node.*

The following result establishes the protocol's consistency guarantee for late joiners. The Stubborn Nakamoto protocol requires an additional assumption to ensure consistency for late joiners: namely, a late joiner must rely on at least one online honest node to maintain consistency with blocks that were finalized before the late joiner woke up.

Theorem 3. *Suppose that node i is a late joiner that wakes up according to Definition 6, and there is some honest node j that is online throughout and did not halt by time t_{on}. Then consistency holds for node i even in the presence of an adversary with unbounded computational power and any number of corrupted nodes.*

Remark 2. *Theorem 3 does not hold under the protocol modification outlined in Remark 1. The reason is that while online nodes can detect and ignore the attack chain described in Section 3.2, late joiners lack sufficient information to select the correct chain after online nodes ignore an attack chain. These challenges are addressed by the recovery oracle, introduced in Section 4.4.*

Proof. Suppose that i is a late joiner, and became online at time t_{on}. Let j be an honest node that is online and did not halt by time t_{on}.

First, let us show that $\mathsf{Log}_i^{t_{on}} = \mathsf{Log}_j^{t_{on}}$. Let $\mathsf{B}_j \in \mathsf{Log}_j^{t_{on}}$ be a block finalized by j. Let t_j denote the time at which block B_j was certified by j and let $\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{j})$ denote the corresponding chain that certified B_j. Because B_j is finalized by j by time t_{on}, we have that $t_j \leq t_{on} - 2\Delta$. Therefore, $\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{j}) \subseteq \mathsf{View}_\mathsf{i}^{\mathsf{t_{on}}}$. If a longest chain in $\mathsf{View}_i^{t_{on}}$ contains B_j, it must be as at least as long as $\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{j}) \subseteq \mathsf{View}_\mathsf{i}^{\mathsf{t_{on}}}$, and therefore include B_j at least k-deep. This implies that $\mathsf{B}_j \in \mathsf{Log}_i^{t_{on}}$, as required.

Otherwise, assume for the sake of contradiction that $\mathsf{Chain}(\bar{\mathsf{B}}')$ is a longest chain in $\mathsf{View}_i^{t_{on}}$ that does not include B_j. Since $\mathsf{Chain}(\bar{\mathsf{B}}')$ is at least as long as $\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{j})$, it must include

some block $\mathsf{B}' \not\sim \mathsf{B}$ that is at least k-deep in $\mathsf{Chain}(\bar{\mathsf{B}}')$. We have that

$$\mathsf{View}_i^{t_{on}-6\Delta} \subseteq \mathsf{View}_j^{t_{on}-5\Delta} \subseteq \mathsf{View}_j^{t_{on}-\Delta} \subseteq \mathsf{View}_i^{t_{on}} ,$$

and therefore $\mathsf{Chain}(\bar{\mathsf{B}}')$ is a longest chain in j's view at time $t_{on} - 5\Delta$. But this implies that node j must halt before time t_{on}, having seen a conflicting certified block.

To show $\mathsf{Log}_i^{t_{on}} \subseteq \mathsf{Log}_j^{t_{on}}$, let $\mathsf{Log}_i^{t_{on}} = \mathsf{Chain}(\mathsf{B}_\mathsf{i})$ and let $\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{i})$ denote a chain that certified B_i. We have that $h(\bar{\mathsf{B}}_i) \geq h(\tilde{B})$ for all $\tilde{\mathsf{B}} \in \mathsf{View}_i^{t_{on}}$. Moreover, because i does not halt by time t_{on}, B_i is k deep in any longest chain in $\mathsf{View}_i^{t_{on}}$. We have that

$$\mathsf{Chain}(\bar{\mathsf{B}}_\mathsf{i}) \subseteq \mathsf{View}_\mathsf{i}^{\mathsf{t_{on}}} = \mathsf{View}_\mathsf{i}^{\mathsf{t_{on}}-6\Delta} \subseteq \mathsf{View}_\mathsf{j}^{\mathsf{t_{on}}-5\Delta} .$$

Therefore, node j certified B_i by time $t_{on} - 5\Delta$. Because j did not halt by time t_{on}, we have that j finalized B_i and every block in $\mathsf{Chain}(\mathsf{B}_\mathsf{i})$ by time t_{on}. Thus, we have that $\mathsf{Log}_i^{t_{on}} = \mathsf{Log}_j^{t_{on}}$.

Now, suppose that node i certifies block B at time $t_1 > t_{on}$ and finalizes it at time $t_2 = t_1 + 2\Delta$. It is immediate that B does not conflict with any block certified or finalized by i. Suppose, for the sake of contradiction, that honest node j finalizes a conflicting block B' at time t_2'. If $t_2' \leq t_{on}$, we have $\mathsf{B}' \in \mathsf{Log}_j^{t_{on}} = \mathsf{Log}_i^{t_{on}}$ by the first part of the proof, and we have a contradiction.

If $t_2' > t_{on}$, it must be that j certifies the block B' at time $t_1' > t_{on} - \Delta$. Let $\mathsf{Chain}(\bar{\mathsf{B}}')$ be the corresponding chain in j's view at time t_1'. We have that $\mathsf{Chain}(\bar{\mathsf{B}}')$ must be longer than any chain in j's view at time t_1', and therefore it is longer than any chain in $\mathsf{View}_i^{t_{on}}$. Node i receives $\mathsf{Chain}(\bar{\mathsf{B}}')$ by time $\tau \leq t_1' + \Delta$. Because $\mathsf{View}_i^{\tau-2\Delta} \subseteq \mathsf{View}_j^{\tau-\Delta} \subseteq \mathsf{View}_j^{t_1'}$, we have that $\mathsf{B}' \in \mathcal{PC}_i^{\tau}(2\Delta)$. Since node i finalized block B, we must have $\tau > t_2$.

Let $\mathsf{Chain}(\bar{\mathsf{B}})$ be the chain in i view that certified B at time t_1. Again, we have that $\mathsf{Chain}(\bar{\mathsf{B}})$ is longer than any chain in $\mathsf{View}_i^{t_{on}}$. Node j receives $\mathsf{Chain}(\bar{\mathsf{B}})$ by time $\nu \leq t_1 + \Delta$. Because $\mathsf{View}_j^{\nu-2\Delta} \subseteq \mathsf{View}_i^{\nu-\Delta} \subseteq \mathsf{View}_i^{t_1}$, we have that $\mathsf{B} \in \mathcal{PC}_j^{\nu}(2\Delta)$. But this implies that $t_1 + \Delta \geq \nu > t_1' + 2\Delta$ and $t_1' + \Delta \geq \tau > t_1 + 2\Delta$, and we have reached a contradiction.

□

When the honest majority assumption holds, late joiners can reliably identify previously finalized blocks that must be in the longest chain in their view. The following theorem provides the protocol's guarantees for late joiners in the absence of an attack.

Theorem 4. *If the honest majority assumption holds, Stubborn Nakamoto satisfies consis-*

tency and liveness for online nodes and late joiners.

Proof. The proof follows from Theorem 2 together with the observation that if consistency and liveness hold for online nodes, then Stubborn Nakamoto and the standard Nakamoto protocol generate identical executions for late joiners (in the sense that any honest node maintains the same finalized ledger). □

4.4 Recovery

We turn to consider the recovery of Stubborn Nakamoto after an attack. An attacker with sufficient resources can cause nodes to halt and the protocol to lose liveness. But suppose that the attacker ends their efforts after some time, and the protocol regains honest majority. Can the protocol recover and regain liveness while maintaining consistency? In particular, can the protocol ensure that any block finalized by a node before or during the attack does not conflict with new blocks that are finalized after recovery?

We show that the Stubborn Nakamoto protocol can recover with external assistance that we model as a *recovery oracle* and that such external assistance is necessary. The recovery oracle models a community response to the attack, allowing us to evaluate the necessary capabilities required from the community. Any protocol requires some community action to establish agreement on its rules. In particular, the Nakamoto protocol relies on an external agreement on a genesis block.[55] We show that Stubborn Nakamoto can recover from an attack with a new genesis block provided by the recovery oracle.

The following defines a recovery oracle that enables the recovery of the Stubborn Nakamoto protocol. This definition formalizes the exact function the community (or another external source) must provide to the protocol. The recovery oracle is practical only if its capabilities are limited to those the community can plausibly implement. While we do not formally model the community's capabilities to achieve consensus, we bear in mind that the community must have been able to initiate the protocol and provide the genesis block (potentially through a costly process) but is unlikely to maintain real-time knowledge of every honest node's local view. Therefore, our definition of the recovery oracle relies solely on having access to the view of a single honest node.

Definition 7 (Recovery Oracle)**.** *A call to a* recovery oracle *at time t updates the protocol's setup by creating a new genesis block B_G, which extends all clear-certified blocks in the view*

[55] That is, the participants of the protocol must achieve agreement on the genesis block by means that are out of protocol.

of some arbitrarily selected honest node. The call also awakes honest nodes that halted.

In other words, a call to the recovery oracle restarts the protocol in a similar manner to the initial launch of the Nakamoto protocol: the call announces a genesis block that any future chain must extend. All honest nodes ignore blocks that do not extend the new genesis block. The recovery oracle requires some contemporary knowledge of the state of honest nodes running the protocol, as it needs to maintain consistency with all finalized blocks. However, the recovery oracle is only required to know the state of a single (arbitrarily selected) honest node. Thus, we argue that this recovery oracle can be plausibly implemented by the community.[56]

Theorem 5 (Consistency and liveness after recovery)**.** *Under synchrony, consider Stubborn Nakomoto with a call to the recovery oracle at time T. Suppose that the network-aware honest majority assumption holds from time T onwards. Then, Stubborn Nakamoto satisfies consistency at any time and liveness after time T.*

An attacker with sufficient resources can cause the Stubborn Nakamoto protocol to lose liveness. Even after a call to the recovery oracle, the attacker can repeat its attack to cause the protocol to lose liveness again. Given the impossibility results mentioned above, such a loss of liveness is unavoidable for any protocol that maintains consistency. However, it is costly for the attacker to continuously violate liveness: each liveness violation requires the attacker to create an attack chain that certifies or potentially certifies a conflicting block, and all blocks in the attack chain are discarded after the call to the recovery oracle. Theorem 5 guarantees that when the attacker stops and Stubborn Nakamoto regains an honest majority, Stubborn Nakamoto recovers liveness (while maintaining consistency throughout).

Remark 3. *Theorem 5 holds under the protocol modification described in Remark 1. If online nodes detect an attack chain and ignore it without halting—while invoking the recovery oracle—then late joiners will be directed to the correct ledger by the recovery oracle, ensuring consistency.*

The following lemma provides the key argument for the consistency of Stubborn Nakamoto after recovery. A genesis block establishes agreement on an initial ledger. Lemma 2 implies that the recovery oracle establishes a ledger that maintains consistency with any transactions finalized by honest nodes before recovery.

[56] The Stubborn Nakamoto protocol is specifically designed to allow a recovery oracle that uses only limited capabilities (that the community can plausibly implement); see Lemma 2 and the discussion below.

Lemma 2 (Consistency guarantees for recovery)**.** *If some honest node i finalizes block B, then every honest node j clear-certifies block B. If some honest node i finalizes block B and does not halt, then all honest nodes finalize block B.*

Proof. Suppose that honest node i certified block B at time t_1 and finalized block B at time $t_2 = t_1 + 2\Delta$. Let $\mathsf{Chain}(\bar{\mathsf{B}})$ be a logest chain in the view of node i at time t such that B is at least k-deep in $\mathsf{Chain}(\bar{\mathsf{B}})$.

Suppose for the sake of contradiction that some honest node j does not clear-certify B. Node j receives $\mathsf{Chain}(\bar{\mathsf{B}})$ by time $t + \Delta$. If $\mathsf{Chain}(\bar{\mathsf{B}})$ (or an extension of it) is a longest chain in the view of node j, then it must be that there is some block B' such that $\mathsf{B}' \not\sim \mathsf{B}$ and $\mathsf{B}' \in \mathcal{C}_j^{t+\Delta}$. But then we have by Lemma 1 that $\mathsf{B}' \in \mathcal{PC}_i^{t+2\Delta}(2\Delta)$, which is a contradiction to i finalizing B.

Suppose then that there is a longer chain $\mathsf{Chain}(\bar{\mathsf{B}}')$ in the view of node j at time $t + \Delta$. Suppose that block B does not appear k-deep in $\mathsf{Chain}(\bar{\mathsf{B}}')$ (otherwise, we reach a contradiction using the previous argument). Then there must be a block B' that is at least k-deep in $\mathsf{Chain}(\bar{\mathsf{B}}')$ and $\mathsf{B}' \not\sim \mathsf{B}$. Again, we have $\mathsf{B}' \in \mathcal{C}_j^{t+\Delta}$, leading to a contradiction.

For the second part of the proof, suppose that honest node i certified block B at time t_1, finalized block B at time $t_2 = t_1 + 2\Delta$, and did not halt at time $t_3 = t_1 + 4\Delta$. By the previous part of the proof, every honest node j clear-certified block B. Suppose for the sake of contradiction that some node j did not finalize block B. Observe that node j certified block B by time $t_1 + \Delta$, and therefore it must be that node j halts without finalizing B at some time $t' \leq t_1 + 3\Delta$. But this implies that there exists some $\mathsf{B}' \in \mathcal{PC}_j^{t'}(2\Delta)$ such that $\mathsf{B}' \not\sim \mathsf{B}$, and by Lemma 1 we have that $\mathsf{B}' \in \mathcal{PC}_i^{t'+\Delta}(4\Delta) \subseteq \mathcal{PC}_i^{t_3}(4\Delta)$ in contradiction to i not halting at time t_3.

□

The proof of Theorem 5 follows from the lemma. A call to the recovery oracle maintains consistency and helps the protocol regain liveness when the attacker no longer controls a majority of the nodes. We remark that the theorem holds even when there are arbitrarily many calls to the recovery oracle.

Proof of Theorem 5. By Theorem 2, the protocol satisfies liveness after it recovers. By Theorem 1, if nodes i, j finalize blocks B, B' of the same height before recovery, then $\mathsf{B} = \mathsf{B}'$. Likewise, by Theorem 1, if nodes i, j finalize blocks B, B' of the same height after recovery, then $\mathsf{B} = \mathsf{B}'$. Suppose that node i finalizes block B before recovery. By Lemma 2, every

honest node clear-certified block B. Therefore, after recovery, the block B is part of the chain of the new genesis block B_G and is finalized by any honest node j. □

The Stubborn Nakamoto with a recovery oracle provides strong guarantees to users. Consistency is always guaranteed, and protocol participants cannot be deceived. An attacker can disrupt liveness, but only temporarily. An attacker can force the community to take potentially costly actions to implement the recovery oracle, but attacks are costly for the attacker (who needs to create blocks that are discarded). Because the existence of a recovery oracle limits the harm an attacker can cause, the recovery oracle may completely deter attackers without ever being used.

The protocol relies on the recovery oracle, but the recovery oracle's influence over the system is highly constrained. First, this dependence is necessary only after the protocol is attacked. Second, any honest node can assume the role of the recovery oracle. Third, the ambiguity that the recovery oracle addresses is minimal, as the protocol ensures agreement on all blocks except those finalized within a brief time window.

4.5 Necessity of the recovery oracle

Can an alternative protocol design achieve the same properties without reliance on an external recovery oracle? Can the functionality of the recovery oracle be provided within the protocol? This section shows that the answers to these questions are negative.

The recovery oracle serves several functions. It resolves some potential ambiguity among online nodes.[57] It resolves potential ambiguity for late-joining nodes. It discards previous conflicting blocks by providing a new genesis block. Last, it informs nodes that halted that they should resume being active.

It may be possible to implement many of these roles within the protocol. Still, without an external recovery oracle, no protocol can guarantee that late-joining nodes adopt the correct ledger, potentially violating consistency for late joiners.

As an illustration, consider a late-joining node i that attempts to join Bitcoin and become a miner. Several similar blockchains use Nakamoto consensus (for example, Dogecoin and Litecoin), and node i must determine which of the chains is the Bitcoin blockchain. Formally, nodes are able to identify the Bitcoin blockchain by using the genesis block provided to it from the protocol's setup.

[57] Under corrupt majority, it is possible that some nodes finalize block B and other nodes halt without finalizing B, and it is possible that some honest nodes certified block B and other honest nodes certified block $\mathsf{B}' \not\sim \mathsf{B}$ (and no node finalizes either block B or block B').

However, there are also different blockchains that share Bitcoin's genesis block and its blockchain up to a certain block. For example, Bitcoin Cash was created by a so-called "hard fork" that split off from Bitcoin and created a new blockchain. The initial portions of the ledgers of Bitcoin and Bitcoin Cash are identical, but they are independent systems, and the later parts of their ledgers diverge. Node i needs more recent information to determine whether it connects to Bitcoin or Bitcoin Cash, as these are separate blockchains that share the same initial genesis block.[58] In particular, the total hash rate of each chain cannot provide the required identification, as node i may be interested in connecting to the chain with a lower hash rate. The recovery oracle allows us to capture the fact that, in practice, users and nodes are able to access resources (external to the protocol) that can identify the two chains.

More generally, the following impossibility result asserts the necessity of the recovery oracle.

Theorem 6 (Necessity of the Recovery Oracle)**.** *Even under synchrony, there is no permissionless protocol that allows late joiners, always satisfies consistency, and for some $\rho > 0$ and $\tau \geq 0$ satisfies liveness after time $T + \tau$ if from time T onwards at least $1 - \rho$ of the nodes are honest.*

The intuition for the proof is that under any permissionless protocol, the attacker can create a parallel copy of the ledger. We show that an attacker can simulate the execution to the point of recovery and bring a late-joining node to adopt a parallel copy that creates a consistency violation. Conceptually, such an attack is similar to a phishing attack which misdirects users to a malicious copy of a website. The attacks on Bitcoin discussed in Section 3 can be seen as instances of this general attack.

Proof. For the sake of contradiction, suppose that such a protocol exists for some $\tau \geq 0$ and $\rho > 0$. We show that an attacker can violate consistency for some late joiners.

By liveness, there is an execution in which all nodes are honest and some honest node i finalizes a ledger Log at some time t. We can extend this execution by having the attacker create a conflicting ledger Log' and send it to all nodes at time $t + 1$. To do so, the attacker runs a parallel copy of the protocol by running $\lceil 1/\rho \rceil$ corrupt nodes for each honest node, where each corrupt node runs the protocol except for not communicating with honest nodes

[58] The hard fork that created Bitcoin Cash included technical changes to the protocol. Bitcoin itself also adopted some technical changes at a later time. We assume that node i arrives after both protocols have made these technical changes, and node i must rely on an external resource to identify the correct chain.

and ignoring messages from honest nodes. This allows the attacker to create a conflicting ledger Log' that is finalized by the corrupt nodes.

Because the protocol is permissionless and the corrupt nodes simulate a different possible execution of the protocol, the ledgers $\mathsf{Log}, \mathsf{Log}'$ are indistinguishable for late joiners. That is, a late joiner cannot know which ledger, Log or Log', has been created by corrupt nodes that do not communicate and ignore messages.

The attacker continues to run $\lceil 1/\rho \rceil$ corrupt nodes for each honest node, where from time $T = t+1$ onwards, each corrupt node runs the protocol wholly and fully communicates with honest nodes (except that all the corrupt nodes have finalized Log' instead of Log). At time T, a late joiner joins the protocol. Observe that for the late joiner, this execution is indistinguishable from an execution in which the corrupt and honest nodes swap roles (that is, honest nodes finalize Log' and corrupt nodes finalize Log) and at most a fraction $1/(1+1/\rho) < \rho$ are corrupt. Thus, at time $T+\tau$, a late joiner believes the protocol regains liveness and finalizes a ledger that extends Log', violating consistency. □

5 Partial Synchrony and Proof-of-Stake

In the years since Nakamoto (2008) showed the possibility of permissionless consensus, there has been extensive work exploring protocol designs. An influential alternative class of protocols is commonly known as Proof-of-Stake (PoS). This class includes a wide range of protocols, which differ in their approach to consensus and the extent to which nodes can freely enter and exit.[59] The distinctive property of PoS protocols is the use of identities and stakes: each protocol node has an identifier and an associated account with a balance that can be controlled by the protocol (commonly called the *stake*).

Establishing node identifiers is helpful for both the consensus protocol and the incentives within the protocol. Identifiers allow the use of classical consensus approaches (for example, the Hotstuff protocol (Yin et al., 2019)), which require nodes to sign messages cryptographically. Signed messages can provide evidence of a node's actions, potentially allowing the protocol to identify faulty nodes (see Sheng et al. 2021, Neu et al. 2022) and punish them by reducing the balances in the node's associated accounts (commonly called *slashing*).

Our framework offers two observations on PoS protocols. First, PoS protocols can deliver better economic security under weaker communication requirements. Second, we present a

[59] See Lewis-Pye and Roughgarden (2023) for a classification of the level of permissionlessness of different PoS protocols.

variation of Theorem 6 for PoS protocols.

5.1 Economic security for a partitioned merchant

The ability to penalize nodes is particularly beneficial when communication is not synchronous.[60] Consider an attacker attempting to deceive a merchant whose node is partitioned from the network. The partitioned merchant's node follows the protocol but only receives messages from the attacker. The merchant's node does not receive messages from other honest nodes in the network. Moreover, assume the merchant is unaware that it is partitioned (i.e., it cannot detect the lack of communication and halt). Assume that the merchant remains partitioned for the duration of the attack but eventually reconnects to the network.[61]

Define the economic security for a partitioned merchant as the attacker's minimal cost of inflicting a consistency violation on a partitioned merchant.

Observe that, under any permissionless protocol, the attacker can deceive a partitioned merchant by creating a conflicting copy of the ledger. To do so, the attacker simulates a parallel execution of the protocol that is indistinguishable from the protocol's correct operation to the partitioned merchant (recall that the merchant cannot communicate with honest nodes and hence cannot detect the existence of another, conflicting ledger). This observation implies an immediate upper bound on the economic security for a partitioned merchant.

Remark 4. *Under a permissionless protocol, the economic security for a partitioned merchant is upper bounded by the cost of simulating an execution of the protocol that leads the merchant's node to finalize a transaction, plus the potential costs of punishments incurred when the merchant reconnects and communicates with other honest nodes.*

Under both Nakamoto and Stubborn Nakamoto, the attacker needs to create a conflicting chain of k blocks to get the partitioned merchant to finalize a transaction and violate

[60] For example, a payment system may want to allow a merchant to accept payment even if they are offline (e.g., a farmers market). However, even traditional payment services may find it challenging to secure offline merchants. For example, the payment processor Square states on its website:

> "There is additional risk when accepting payments offline. Square is not responsible for any loss due to declined cards or expired payments taken while offline."

(https://squareup.com/help/us/en/article/7777-process-card-payments-with-offline-mode, accessed July 2024)

[61] Such partitioned merchant is possible under both the asynchronous communication model and the partial synchrony communication model (Dwork et al., 1988), as the attacker may delay the delivery of messages (both to and from the merchant) for an arbitrary (but finite) length of time.

consistency. In the rental model, the attacker's cost of creating k blocks is $k \cdot p_{\mathsf{B}}$. The two protocols differ in the response of honest nodes after the partition ends. Under Nakamoto, the attacker can induce all honest nodes to adopt its conflicting ledger, enabling the attacker to collect block rewards and recoup its costs (as described in Section 3.2). By contrast, under Stubborn Nakamoto, the net cost to the attacker remains $k \cdot p_{\mathsf{B}}$. The attacker can deceive a partitioned merchant, but conflicting blocks cannot become part of the consensus chain. Note that under Stubborn Nakamoto, nodes are anonymous, and there is no additional punishment for creating a conflicting chain.

Under many PoS protocols, the cost of simulating a (conflicting) execution of the protocol is negligible. Instead, the protocol may be able to detect and penalize nodes that create a conflicting ledger (for example, by witnessing the node's signature on the conflicting ledger). In particular, if the protocol can detect nodes that have deviated from the protocol, it may punish only the deviating nodes and thus avoid the tragedy of the commons[62] described in Section 3.4.

To establish an upper bound, consider a hypothetical PoS protocol that can perfectly identify all the nodes that create a conflicting ledger (and assume that the cost of protocol execution is 0). The maximal possible punishment that can be imposed on a node is the slashing of its entire stake (clearing the balance in the node's associated account). Thus, an upper bound for the attacker's total cost is the entire stake of all the nodes that created the conflicting ledger. In other words, the economic security for a partitioned merchant is bounded by the total amount of stake required to create an execution of the protocol that finalizes a transaction.

A simple comparison suggests that punishment in PoS can lead to much greater economic security for a partitioned merchant than the cost of execution in Stubborn Nakamoto. If the Stubborn Nakamoto protocol spends p_{B} per block, under the rental model, it achieves economic security for a partitioned merchant of $k \cdot p_{\mathsf{B}}$. A PoS protocol can potentially achieve greater economic security by requiring nodes to stake. Suppose that the cost of executing the protocol is negligible, and the protocol only needs to compensate nodes for forgone interest. In that case, block rewards should equal forgone interest on the total stake, and nodes provide a total stake of p_{B}/r, where r is the interest rate per block.[63] If the attacker suffers losses equal to a fraction λ of the total stake, the attacker's cost is $\lambda p_{\mathsf{B}}/r$.

[62] To completely avoid the tragedy of the commons, the protocol should punish deviating nodes even when there is no successful attack.

[63] The protocol can avoid these costs if it can invest the staked funds. We thank Niels Gormsen for this observation.

However, note that PoS protocols face several challenges that are omitted from the above discussion. It may be challenging to detect deviating nodes, and the protocol may detect only some of the deviating nodes (Sheng et al., 2021, Neu et al., 2022). If the protocol mistakenly slashes honest nodes, it must compensate all nodes for this potential harm. Moreover, the attacker may cause further deviations from the protocol to prevent the protocol from enacting the punishment (Budish et al., 2024).

5.2 Necessity of Recovery Oracles

We provide a version of the impossibility result given in Theorem 6 for PoS protocols. Consider a PoS protocol that guarantees consistency. Concurrent work of Sridhar et al. (2023) shows the feasibility of such PoS protocols under synchronous communication. But no protocol can always guarantee consistency and liveness under synchronous communication with late joiners (Pass and Shi, 2017a, Lewis-Pye and Roughgarden, 2023).[64] Can a PoS protocol only lose liveness temporarily, halting progress while there is no honest majority, and regain liveness once honest majority is restored?

By Theorem 6, a recovery oracle is necessary if the protocol is permissionless. In practice, many PoS protocols require that active nodes approve late-joining nodes.[65] This required approval can help secure the protocol by limiting the attacker's ability to simulate a conflicting protocol execution. However, such a limitation is helpful only if it introduces a different problem: the protocol can get stuck if no late-joining nodes are allowed to enter. We state this tension in the following corollary.

Corollary 1. *Consider a protocol that, under synchrony, allows late joiners, always satisfies consistency, and for some $\rho > 0$ and $\tau \geq 0$ satisfies liveness after time $T + \tau$ if from time T onwards at least a fraction $1 - \rho$ of the nodes are honest.*

Then there exists some $\gamma \leq 1$ such that if a fraction γ of the nodes crash, the protocol loses liveness forever.

The proof follows a similar argument to the proof of Theorem 6.

[64] If all nodes are online and communication is synchronous, then the Dolev–Strong protocol guarantees consistency and liveness even without honest majority. But honest majority is necessary for the consistency of the Dolev–Strong protocol when late joiners are allowed.

[65] For example, a node may be required to issue a transaction that establishes its stake before participating in the protocol. If such transactions are written to the ledger, and active nodes must process them (and can censor them), late-joining nodes will require approval from active nodes.

Proof. Suppose, for the sake of contradiction, that such a protocol exists and regains liveness at some time after all nodes crash at time t. Then, it must be possible for late-joining nodes to join the protocol at some time $\tau > t$ despite all previous nodes crashing. Similarly to the proof of Theorem 6, the attacker can exploit this possibility to create a conflicting ledger Log' by launching a separate collection of nodes that follow the protocol, except that they run as if all nodes crashed at time t.

Alternatively, suppose that the attacker controls all existing nodes at time t. The attacker continues running the nodes but makes it appear as if all nodes crashed. New late-joining honest nodes enter the protocol and generate the ledger Log. Because a late joiner cannot distinguish between the two scenarios, the protocol cannot guarantee consistency for late joiners. □

Corollary 1 and Theorem 6 show that any protocol must require some external help under some scenarios: either to recover from attacks or to recover after all active nodes halt. In other words, some dependency on the community is unavoidable. Still, protocols can minimize this dependency by relying on external help only in rare scenarios and reducing the requirements needed for implementing such external help.

6 Discussion

Our results demonstrate the feasibility of an open-source record-keeping service that operates without a trusted record keeper. While users may not be concerned with the internal structure of the system, they can observe its beneficial properties: it protects its users from deception, allows for free entry and exit of operators, and protects users from monopoly's harm.

Central to our design is prioritizing consistency (security) over liveness (timely ledger updates). Roughly speaking, the Stubborn Nakamoto protocol either completely avoids the consistency violations of the Nakamoto protocol or converts them into liveness violations. We argue that financial applications should prioritize consistency over liveness. Recording a transaction in a financial ledger is beneficial only if other parties recognize that record. If the ledger is updated with records that are not recognized or are later invalidated, users do not receive any benefit and are exposed to deception and financial loss. Another justification for prioritizing consistency is that an attacker can clearly benefit from consistency violations that enable deception. By contrast, liveness violations do not offer the attacker such direct gains.

Many questions about the design of payment systems are beyond the scope of this paper. The Stubborn Nakamoto protocol provides a general ledger, enabling users to record arbitrary data. Bitcoin is one example of such a payment system. The technology allows for many others. For example, the ledger could be used to record balances in USD or document land ownership. The system may allow users to be pseudo-anonymous, or can require that every user is fully identifiable. Such design choices involve multiple significant issues, including privacy concerns and anti-money laundering challenges. For a discussion of such challenges, see, for example, Rogoff (2017) and Makarov and Schoar (2022).

Finally, one significant benefit of open-source technology is its inherent openness and modularity. Much like how anyone can join a permissionless protocol as a node, open-source frameworks enable developers to build upon existing services. This approach encourages innovation and competition in the market for higher-level services. For instance, while foundational Internet protocols such as TCP/IP may not be of much use to end users alone, they establish a crucial infrastructure enabling user-friendly commercial applications. Similarly, even permissionless protocols that provide limited functionality can be beneficial as foundational rails that allow firms to build more complex services, fostering a competitive market where diverse and innovative services can thrive.

References

Abadi, J. and Brunnermeier, M. (2022). Blockchain economics. *Working Paper*.

Abraham, I., Dolev, D., Gonen, R., and Halpern, J. (2006). Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 53–62.

Admati, A. and Hellwig, M. (2014). *The bankers' new clothes: What's wrong with banking and what to do about it*. Princeton University Press.

Auer, R. (2019). Beyond the doomsday economics of 'proof-of-work' in cryptocurrencies. *BIS working paper*.

Badertscher, C., Lu, Y., and Zikas, V. (2021). A rational protocol treatment of 51% attacks. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Confer-*

ence, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III 41, pages 3–32. Springer.

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

Bonneau, J. (2016). 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, E. (2025). Trust at scale: The economic limits of cryptocurrencies and blockchains. *The Quarterly Journal of Economics*, 140(1):1–62.

Budish, E., Lewis-Pye, A., and Roughgarden, T. (2024). The economic limits of permissionless consensus. *arXiv preprint arXiv:2405.09173*.

Buterin, V. (2016). A proof of stake design philosophy. Blog post, https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51. Accessed: Feb 2025.

Buterin, V. (2023). Don't overload ethereum's consensus. https://vitalik.eth.limo/general/2023/05/21/dont_overload.html. Blog post [Online; accessed 25-Feb-2025].

Buterin, V. and Griffith, V. (2017). Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*.

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

Cai, L., Liu, J., Weinberg, S. M., and Zhou, C. (2024). Profitable manipulations of cryptographic self-selection are statistically detectable. *arXiv preprint arXiv:2407.16949*.

Carlsten, M., Kalodner, H., Weinberg, S. M., and Narayanan, A. (2016). On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 154–167.

Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186.

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.

CNBC (2024). Buyers during berkshire price glitch can't keep their big bargains. https://link.cnbc.com/public/35649502. [Online; accessed 25-Feb-2025].

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

Dixon, C. (2024). *Read Write Own: Building the Next Era of the Internet*. Random House.

DOJ (2010). U.s. and plaintiff states v. american express co., et al., no. 10-4496 (e.d.n.y. 2010). United States Department of Justice, Antitrust Division https://www.justice.gov/atr/case/us-and-plaintiff-states-v-american-express-co-et-al. Accessed: 2024-09-09.

DOJ (2020). DOJ blocks Visa's acquisition of Plaid. https://www.justice.gov/opa/pr/justice-department-sues-block-visas-proposed-acquisition-plaid. Accessed: 2024-09-09.

Dolev, D. and Strong, H. R. (1983). Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323.

Ethereum Classic (2022). Code is law. https://ethereumclassic.org/why-classic/code-is-law. Web page [Online; accessed 25-Feb-2025].

Ethereum Foundation (2016). Hard fork completed. https://blog.ethereum.org/2016/07/20/hard-fork-completed. Blog post [Online; accessed 25-Feb-2025].

Ethereum Foundation (2023). Weak subjectivity. Ethereum Blog, https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/weak-subjectivity/.

European Parliament and Council (2015). Regulation (eu) 2015/751 on interchange fees for card-based payment transactions. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32015R0751. Accessed: 2024-09-09.

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

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382.

Gans, J. S. and Halaburda, H. (2024). "zero cost" majority attacks on permissionless proof of work blockchains. *Management Science*.

Garay, J., Katz, J., Maurer, U., Tackmann, B., and Zikas, V. (2013). Rational protocol design: Cryptography against incentive-driven adversaries. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 648–657. IEEE.

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, R. J. and van Oordt, M. R. (2023). Why fixed costs matter for proof-of-work–based cryptocurrencies. *Management Science*, 69(11):6482–6507.

Gonczarowski, Y. A. and Moses, Y. (2023). Common knowledge, regained. *arXiv preprint arXiv:2311.04374*.

Greenwood, R. and Scharfstein, D. (2013). The growth of finance. *Journal of Economic perspectives*, 27(2):3–28.

Guiso, L., Sapienza, P., and Zingales, L. (2004). The role of social capital in financial development. *American economic review*, 94(3):526–556.

Guiso, L., Sapienza, P., and Zingales, L. (2008). Trusting the stock market. *the Journal of Finance*, 63(6):2557–2600.

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

Halaburda, H., Haeringer, G., Gans, J., and Gandal, N. (2022). The microeconomics of cryptocurrencies. *Journal of Economic Literature*, 60(3):971–1013.

Halaburda, H., He, Z., and Li, J. (2021). An economic model of consensus on distributed ledgers. Technical report, National Bureau of Economic Research.

Hayashi, F. and Maniff, J. L. (2019). Public authority involvement in payment card markets: Various countries–august 2019 update. *Federal Reserve Bank of Kansas City*.

Herkenhoff, K. F. and Raveendranathan, G. (2020). Who bears the welfare costs of monopoly? the case of the credit card industry. Technical report, National Bureau of Economic Research.

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

Kiayias, A., Koutsoupias, E., Kyropoulou, M., and Tselekounis, Y. (2016). Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 365–382.

Kocherlakota, N. R. (1998). Money is memory. *journal of economic theory*, 81(2):232–251.

La Porta, R., Lopez-de Silanes, F., and Shleifer, A. (2006). What works in securities laws? *The journal of finance*, 61(1):1–32.

Lamport, L. (1998). The part-time parliament. *ACM Trans. on Computer Systems*, 16:133–169.

Lamport, L. (2001). Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58.

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

Levine, M. (2021). Citi can't have its $900 million back. https://www.bloomberg.com/opinion/articles/2021-02-17/citi-can-t-have-its-900-million-back. Bloomberg Opinion [Online; accessed 25-Feb-2025].

Levine, M. (2022a). Citi got its $500 million back. https://www.bloomberg.com/opinion/articles/2022-09-12/citi-got-its-500-million-back. Bloomberg Opinion [Online; accessed 25-Feb-2025].

Levine, M. (2022b). Matt levine's money stuff: Citi and revlon are in murky waters. https://news.bloomberglaw.com/mergers-and-acquisitions/matt-levines-money-stuff-citi-and-revlon-are-in-murky-waters. Bloomberg Opinion [Online; accessed 25-Feb-2025].

Lewis-Pye, A. and Roughgarden, T. (2023). Permissionless consensus. *arXiv preprint arXiv:2304.14701*.

Makarov, I. and Schoar, A. (2022). Cryptocurrencies and decentralized finance (defi). *Brookings Papers on Economic Activity*, 2022(1):141–215.

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 & Company (2024). Global payments in 2024: Simpler interfaces, complex reality. https://www.mckinsey.com/industries/financial-services/our-insights/global-payments-in-2024-simpler-interfaces-complex-reality#/. [Online; accessed 25-Feb-2025].

Morningstar (2024). Visa inc class a analysis. Economic Moat by Brett Horn, Updated June 13, 2024.

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, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.pdf.

Nesbitt, M. (2019). Deep chain reorganization detected on ethereum classic (etc). Coinbase Blog, https://www.coinbase.com/blog/deep-chain-reorganization-detected-on-ethereum-classic-etc. Online [accessed 25-Feb-2025].

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, 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.

Newman, S. H. (2023). Decentralization cheapens corruptive majority attacks. In *5th Conference on Advances in Financial Technologies*.

Pagnotta, E. S. (2022). Decentralizing money: Bitcoin prices and blockchain security. *The Review of Financial Studies*, 35(2):866–907.

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.

Pass, R. and Shi, E. (2017a). Rethinking large-scale consensus. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 115–129. IEEE.

Pass, R. and Shi, E. (2017b). The sleepy model of consensus. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*, pages 380–409. Springer.

Pease, M., Shostak, R., and Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234.

Philippon, T. (2015). Has the us finance industry become less efficient? on the theory and measurement of financial intermediation. *American Economic Review*, 105(4):1408–1438.

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

Rajan, R. and Zingales, L. (1998). Financial dependence and growth. *American Economic Review*, 88(3):559–86.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Rogoff, K. (2017). *The curse of cash: How large-denomination bills aid crime and tax evasion and constrain monetary policy*. Princeton University Press.

Rosenbaum, A., Baughman, G., Manuszak, M. D., Stewart, K., Hayashi, F., and Stavins, J. (2017). Faster payments: market structure and policy considerations. *Federal Reserve Bank of Kansas City Working Paper No. RWP*, pages 17–14.

Sankagiri, S., Wang, X., Kannan, S., and Viswanath, P. (2021). Blockchain cap theorem allows user-dependent adaptivity and finality. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 84–103. Springer.

Schumpeter, J. A. (1911). A theory of economic development. *Harvard University Press*.

Shanaev, S., Shuraeva, A., Vasenin, M., and Kuznetsov, M. (2019). Cryptocurrency value and 51% attacks: evidence from event studies. *The Journal of Alternative Investments*, 22(3):65–77.

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, E. (2022). Foundations of distributed consensus and blockchains. Textbook, https://www.distributedconsensus.net/.

Sridhar, S., Tas, E. N., Neu, J., Zindros, D., and Tse, D. (2024). Consensus under adversary majority done right. *arXiv preprint arXiv:2411.01689*.

Sridhar, S., Zindros, D., and Tse, D. (2023). Better safe than sorry: Recovering after adversarial majority. *arXiv preprint arXiv:2310.06338*.

Tabarrok, A. (2019). Bitcoin is less secure than most people think. *https://marginalrevolution.com/marginalrevolution/2019/01/bitcoin-much-less-secure-people-think.html, Retrieved Mar, 2024*.

The New York Times (2024). Visa antitrust lawsuit over debit cards. https://www.nytimes.com/2024/09/24/business/dealbook/visa-antitrust-lawsuit-debit-cards.html. [Online; accessed 25-Feb-2025].

United States Cong. House Committee on Financial Services Task Force on Financial Technology (2020). Is cash still king? reviewing the rise of mobile payments. 116th Cong. 2nd sess. Testimony of Aaron Klein, Fellow, Economic Studies, Brookings Institution.

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.

Wired (2019). Blockchain can wrest the internet from corporations' grasp. https://www.wired.com/story/how-blockchain-can-wrest-the-internet-from-corporations/. [Online; accessed 25-Feb-2025].

Wired (2022). Paradise at the crypto arcade: Inside the web3 revolution. https://www.wired.com/story/web3-paradise-crypto-arcade/. [Online; accessed 25-Feb-2025].

Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G., and Abraham, I. (2019). Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356.

A Implementing a Payment System Using a Public Ledger

This appendix defines the functionality of a basic payment system and demonstrates how it can be implemented using a public ledger and standard cryptographic tools. In other words, it shows that an open ledger satisfying economic security (consistency) and liveness is sufficient for implementing a payment system.

Although the payment system presented here is basic, it establishes the infrastructure needed for more sophisticated services. For example, a payment provider can offer more customer friendly interfaces by managing accounts and processing transactions on behalf of customers.

Users can also receive additional protection via courts and the legal system, just as they would under traditional payment systems. Cash (physical coins or notes) can be stolen, as there are no built-in mechanism preventing a thief from using stolen notes. Protection against theft comes from courts and the rule of law, which exist outside the payment system itself. Similarly, while the basic payment system described here may not prevent coerced or erroneous payments, legal recourse can mandate the return of funds. One illustrative example is Citibank's struggle to reverse an erroneous $900 million payment, which was extensively by Matt Levine (Levine, 2021, 2022a,b).

A.1 Definition of a payment system

A payment system allows users to maintain balances and initiate valid transactions that modify those balances. A transaction $x \xrightarrow{s} y$ deducts the amount s from the balance of account x and adds the same amount s to the balance of account y. A transaction $x \xrightarrow{s} y$ is valid if the following two conditions are satisfied:

1. The owner of account x approves the transfer of funds from the account.

2. Account x has a sufficient balance of at least s.

For example, physical coins implement a basic payment system in which a user's balance is simply the number of coins in their possession. To transfer funds, a user hands over coins to the recipient. This transfer is only feasible if the user has enough coins, and standard property protections typically ensure the owner's approval for a transaction.

Our definition omits certain elements of a payment system. There may be a special entity that issues new coins. A special entity may have the authority to modify balances or reverse transactions. A special entity might vet prospective users before allowing them to open an account. Each of these elements can be implemented by combining a public ledger with a special entity that is conferred such powers.

A.2 Implementing a payment system using a public ledger

For illustration, consider a metaphorical village with a large stone in its town square, where anyone can write new information and everyone can read all of the information inscribed on it. Moreover, everyone observes the same recorded data. In other words, this stone functions as a public ledger that satisfies consistency and liveness.

To execute a transaction $x \xrightarrow{s} y$, the owner of account x appends the message $\big(x, y, s, \sigma\big)$ to the end of the stone ledger. The message specifies the inteded transaction and includes the cryptographic signature $\sigma = \sigma(x, y, s)$ that proves the transaction was authorized by the owner of account (Rivest et al., 1978).

The stone ledger forms a list of all transactions. To calculate balances, each user reads the ledger and tallies changes resulting from valid transactions, assuming an agreed-upon set of initial balances. Any invalid transactions written to the stone are ignored by readers.

The ledger recorded on the stone is a list of all transactions. To calculate balances, users read the stone and tally the balances as they change with all valid transactions (assuming

some agreed upon initial account balances). If someone wrote invalid transactions to the stone, the readers can ignore the invalid information.

This stone ledger implements a payment system:

- Cryptographic signatures ensure that each transaction is approved by its owner, satisfying condition 1.
- The public ledger enables every reader to verify that no transaction exceeds the sender's balance, satisfying condition 2.

B Distributed Consensus Formal Execution Model

The proof of Theorem 2 uses a similar approach to the proof of consistency and liveness of the Nakamoto protocol of Pass et al. (2017), Shi (2022). The proof requires a few additional definitions and utilizes several stochastic bounds that were proved in earlier works (Pass et al., 2017, Shi, 2022). We state these here.

B.1 Execution model

We introduce some notation regarding the protocol's execution model in order to formally state the conditions under which Nakamoto has provable guarantees.

We assume that n nodes participate in the consensus protocol, all with equal mining power.[66] Time is discrete and divided into rounds. Each round corresponds to one calculation of a hash by a miner. 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.

B.2 Definitions

Because the Nakamoto consensus protocol requires randomness, a small probability of failure may be inevitable. The formal definition of liveness for the Nakamoto and Stubborn Nakamoto protocols bounds the failure probability by a security parameter λ. The protocol can tune its security parameter λ by selecting a required block depth k, such that $k = \omega(\log \lambda)$.

[66] A node with larger mining power can be viewed as multiple nodes.

For the sake of simplicity, we assume that an honest node that mines a block can include all the outstanding valid transactions (not included in the longest chain) given its local view. Let $T_{\text{conf}} = T_{\text{conf}}(\lambda, \Delta, n, \rho)$ be a function of the security parameter λ, network delay Δ, number of nodes n, and fraction of adversary nodes ρ. We call T_{conf} the *confirmation time*.

Definition 8 (T_{conf}-Liveness)**.** *The protocol satisfies T_{conf}-Liveness if there is some negligible*[67] *function* $\mathsf{negl}(\cdot)$ *such that with* $1 - \mathsf{negl}(\lambda)$ *probability when some honest node receives a transaction* tx *at some time t, then by time $t + T_{\text{conf}}$ all honest nodes' finalized ledgers include* tx.

We can now formally state the network-aware honest majority assumption used in Theorem 2.

Definition 9 (Network-Aware Honest Majority)**.** *Consider a polynomial length execution of Stubborn Nakamoto with a required block depth k that is super-logarithmic function in the desired security parameter λ. Let $\nu = 2pn\Delta$. The Network-Aware Honest Majority assumption is satisfied if $\nu < 1/2$ and there exists some (arbitrarily small) constant $\phi \in (0, 1)$ such that*

$$(1 - \rho)(1 - \nu) \geq (1 + \phi)\rho \; .$$

The term $(1-\rho)(1-\nu)$ can be viewed as the fraction of honest mining power discounted by the network delay Δ. Therefore, this assumption requires that the honest mining power, even after accounting for network delay, exceeds the corrupt mining power by a small constant margin.

B.3 Stochastic bounds

The following stochastic bounds proved in earlier works (Pass et al., 2017, Shi, 2022) are used in the proof of Theorem 2. We state them here for completeness.

Define convergence opportunity in the same way as Section 17.3 of Shi (2022). Specifically, round 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. Given a run of Nakamoto or Stubborn Nakamoto, 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[68] mined

[67] We say that $\mathsf{negl} : \mathbb{R} \to \mathbb{R}_+$ is a negligible function iff for any fixed polynomial function $p(\lambda)$, there exists λ_0 such that for any $\lambda > \lambda_0$, $\mathsf{negl}(\lambda) < 1/p(\lambda)$. In other words, a negligible function is one that vanishes faster than any inverse-polynomial function.

[68] That is, blocks mined by nodes that are not honest nodes.

between t and t'. Let $\alpha := p \cdot (1-\rho)n$ denote the expected number of honest nodes that mine a block in a given round, and let $\beta := p \cdot \rho n$ denote the expected number of corrupt nodes that mine a block in each round.

Lemma B.1 (Lower bound on convergence opportunities (Pass et al., 2017, Shi, 2022)). *For any positive constant $\varepsilon > 0$ 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*

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

Lemma B.2 (Upper bound on adversarially mined blocks (Pass et al., 2017, Shi, 2022)). *For any constant $0 < \varepsilon < 1$, 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 $t_0, t_1 \geq 0$ such that $t := t_1 - t_0 > k/\beta$, we have that*

$$A[t_0 : t_1] \leq (1+\varepsilon)\beta t \ .$$

for any $t \geq k/\beta$. That is, the number of adversarially mined blocks in any t-sized window is upper bounded by $(1+\epsilon)\beta t$.

Lemma B.3 (Total block upper bound (Pass et al., 2017, Shi, 2022)). *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 t_0 and t_1 such that $np(t_1 - t_0) \geq k$, the total number of blocks successfully mined during $(t_0, t_1]$ by all nodes (honest and corrupt alike) is upper bounded by $(1+\varepsilon)np(t_1 - t_0)$.*

B.4 Proof of Theorem 2

Proof of Theorem 2. Given an execution of the Stubborn Nakamoto protocol, we can construct an execution of the Nakamoto[69] protocol. If the two executions are identical, in the sense that any honest node maintains the same finalized ledger at any point during the run, then Stubborn Nakamoto's properties follow from those of Nakamoto. We show that under the network-aware honest majority assumption, the two executions are identical except for

[69] We use the same confirmation depth k as our Stubborn Nakamoto for the execution of the Nakamoto protocol.

a negligible probability, and therefore the T_{conf}-Liveness of Stubborn Nakamoto follows from the T_{conf}-Liveness and consistency of Nakamoto (Pass et al., 2017, Shi, 2022).

The reason the executions may differ is that an honest node i may halt under Stubborn Nakamoto. Consider the first round t_1 such that some honest node i certifies a block B and by round $t_1 + 4\Delta$ node i halts having seen a 4Δ-potentially certified block B' that conflicts with B. (That is, node i halted at either steps 1, 2 or 3)

Let $\mathsf{Chain}(\bar{\mathsf{B}})$ be the chain in node i's local view in round t_1 containing B at least k-deep. Let $\mathsf{B}' \in \mathcal{PC}_i^{t_1+4\Delta}(4\Delta)$ be a block that conflicts with B, and let $t_1 \leq t \leq t_1 + 4\Delta$ be the fist time at which $\mathsf{B}' \in \mathcal{PC}_i^t(4\Delta)$. Let $\mathsf{Chain}(\bar{\mathsf{B}}')$ be the chain in node i's local record such that $h(\bar{\mathsf{B}}') - h(\mathsf{B}') + 1 \geq k$ and $h(\bar{\mathsf{B}}') \geq h(\tilde{\mathsf{B}})$ for any $\tilde{\mathsf{B}} \in \mathsf{View}_i^{t-4\Delta}$. That is, block B' is at least k-deep in the chain $\mathsf{Chain}(\bar{\mathsf{B}}')$, and the chain $\mathsf{Chain}(\bar{\mathsf{B}}')$ is longer than any chain in the view of node i as of $t - 4\Delta$.

Let $\mathsf{B}_{\text{common}}$ the last block shared in common by $\mathsf{Chain}(\bar{\mathsf{B}})$ and $\mathsf{Chain}(\bar{\mathsf{B}}')$. Suppose that $\mathsf{B}_{\text{common}}$ is mined in round $s - 1$. It holds that all blocks in $\mathsf{Chain}(\bar{\mathsf{B}})$ and $\mathsf{Chain}(\bar{\mathsf{B}}')$ after $\mathsf{B}_{\text{common}}$ are mined in round s or later. Let $\tau = t - s$. Because the chains $\mathsf{Chain}(\bar{\mathsf{B}})$ and $\mathsf{Chain}(\bar{\mathsf{B}}')$ differ by at least k blocks, by total block upper bound (see Lemma B.3), it must be that $\tau > \frac{k}{2pn}$.

Observe that it must be that $\mathbf{C}[s : t - 5\Delta] \leq \mathbf{A}[s : t]$. Otherwise, there would have been a block B mined by an honest node during a convergence opportunity within $[s, t - 4\Delta - \Delta]$, and that block B would have to appear in both $\mathsf{Chain}(\bar{\mathsf{B}})$ and $\mathsf{Chain}(\bar{\mathsf{B}})$ (see Section 17.3 of Shi 2022). Therefore, it suffices to prove that, except with negligible λ probability, it must be that $\mathbf{C}[s : t - 5\Delta] > \mathbf{A}[s : t]$.

Let there be some $\varepsilon, \varepsilon' \in (0, 1)$. By the lower bound on convergence opportunity (see Lemma B.1), except with negligible probability, we have that

$$\mathbf{C}[s : t - 5\Delta] > (1 - \varepsilon)(1 - 2pn\Delta)\alpha(\tau - 5\Delta) \ ,$$

where $\alpha := p \cdot (1 - \rho)n$ denotes the expected number of honest nodes that mine a block in a given round. By adversarial block upper bound, (see Lemma B.2) we have that for any positive constant ε_a, it holds that

$$\mathbf{A}[s : t] < (1 + \epsilon_a)\beta\tau \ ,$$

where $\beta = p \cdot \rho n$ denotes the expected number of corrupt nodes that mine a block in each round.

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

$$\begin{aligned}
\mathbf{C}[s : t - 5\Delta] &> (1 - \varepsilon)(1 - \nu)\alpha(\tau - 5\Delta) \\
&> (1 - \varepsilon_1)(1 - \nu)\alpha\tau \\
&> (1 - \varepsilon_1)(1 + \phi)\beta\tau \\
&> (1 + \varepsilon')(1 + \frac{\phi}{2})\beta\tau \\
&> (1 + \varepsilon')\beta\tau
\end{aligned}$$

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

This implies that except with negligible in λ probability we have that $\mathbf{C}[s : t - 5\Delta] > \mathbf{A}[s : t]$, completing the proof.

□

C Other Omitted Proofs

Proof of Proposition 1. Suppose that 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 $(\mathsf{B}_0, \ldots, \mathsf{B}_{n-1}, \mathsf{B}_n, \ldots, \mathsf{B}_{n+K})$.

At time t, the attacker mines the attack chain $(\mathsf{B}_0, \ldots, \mathsf{B}_{n-1}, \mathsf{B}'_n, \ldots, \mathsf{B}'_{n+L})$ for some $L > K$ and communicates it immediately to all other miners. If the attack chain is the longest chain known to any miner, it becomes the consensus chain and is adopted by all honest miners.

The attacker pays $D \cdot c$ in expectation per mined block. If honest miners adopt the attack chain, 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 $p_{\mathsf{B}} \geq D \cdot c$, 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 arbitrarily close to one if the attacker computes the same number of hashes over a shorter time. □

D Examples of Profitable Consistency Violations

Attackers can profit from consistency violations by deceiving and taking advantage of a counterparty. 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 party can detect which transaction is valid. Conversly, if the attacker can deceive a merchant who follows the protocol into accepting a transaction that conflicts with the consensus ledger, then consistency is violated.

History rewrite Consistency requires that any node's local ledger agrees with any previously finalized data. In particular, it implies that the ledger cannot revert finalized transactions. A dishonest attacker can gain if they can undo a finalized transaction. For example, consider a transaction that transfers ownership of an asset from the attacker to a merchant. If the price of the asset increases after the sale, the attacker can benefit from reverting the transaction and retroactively canceling the transfer of ownership.

E Other Factors Deterring Attacks

This section explores additional factors that deter attacks. Despite numerous attacks on cryptocurrencies using the Nakamoto protocol, major cryptocurrencies such as Bitcoin and Ethereum have remained secure. The rental model and the bribery model discussed in Section 3 fail to explain why attacks on these protocols are not commonplace. This indicates that, in practice, other factors contribute to deterring attacks. We discuss such factors and identify elements that can be incorporated into protocol design to enhance security.

We categorize these factors into three groups. First is the user community's reaction to an attack on the protocol. The second category encompasses economic frictions that hinder the execution of an attack. The third category includes reasons for the system's security that suggest that the system is secure because it is not in fact decentralized. Altogether, the discussion suggests that Bitcoin's economic security comes not from the cost of mining but rather from these external deterrents described below.

Community response Even if the attack is not detectable within the protocol, the community can detect an attack. Much of the literature (e.g., Budish 2025, Auer 2019, Garratt and van Oordt 2023) considers that an attack will trigger a decline in the cryptocurrency exchange rate. This assumption inherently relies on the public's awareness that an attack has occurred.

In practice, the community can identify the attack fork and the honest fork under the attack described in 3.2. As described in Section 3, in the view of an honest node, an attack appears to be a reconciliation after a network partition. That is, an honest node will receive the blocks in the attack fork significantly later than blocks of comparable height in the honest fork.[70] Any honest node active during the attack can recognize the attack chain as either a deliberate attack or a highly improbable event of partition reconciliation.

The community can collectively decide to undo the attack. If the community agrees that the miners were attacked and the ledger was changed illegally, the community can direct all miners to follow with the correct honest fork and ignore the attack fork. Doing so reverts any effect of the attack, leaving the ledger unchanged by the attack. The community response poses a significant risk to the attacker, who is left to pay all costs of the attack, but without the attack having any eventual effect on the ledger.

While community-driven interventions in blockchain networks are rare, they are not without precedent. A notable instance is the Ethereum community's decision to amend the Ethereum ledger in response to the DAO attack.[71] Such measures, however, necessitate a community that possesses both the capability and the willingness to undertake manual interventions. For example, Ethereum Classic rejects such manual interventions.[72] Interestingly, Ethereum did not suffer any consistency attacks, whereas Ethereum Classic suffered from multiple attacks.[73]

Note that a community response can deter attacks even if the community can only detect an attack (without agreeing on the correct ledger that should be restored). As discussed in Section 3.4, preventing the attacker from collecting the block rewards after the attack lowers the payment to attackers relative to honest miners (when there is no attack). This creates

[70] The attacker must keep hidden the blocks in the attack chain $(\mathsf{B}'_n, \ldots, \mathsf{B}'_{n+L})$ until the merchant finalizes the block B_n (otherwise, the merchant will not finalize B_n). Thus, an honest miner receives block B'_n only after receiving block B_{n+k}, which it receives significantly later than block B_n. Recall that honest nodes mine conflicting blocks $\mathsf{B}_n, \mathsf{B}'_n$ only if these blocks are mined simultaneously (up to network delays).

[71] See, for example, https://blog.ethereum.org/2016/07/20/hard-fork-completed, accessed February 2024.

[72] https://ethereumclassic.org/why-classic/code-is-law

[73] https://www.coinbase.com/blog/deep-chain-reorganization-detected-on-ethereum-classic-etc, accessed February 2024.

a penalty for an attack that generates some detterance.

Frictions In practice, it may be difficult to operationalize an attack. For example, even if a bribery contract is financially beneficial for miners, it may be difficult to inform and recruit miners. Moreover, miners may not be fully profit-driven, or may be altruistic. This poses a risk to the attacker. A bribery attack needs to commit to paying miners who mine blocks in the attack chain. If the bribery attack fails to solicit enough mining power to cooperate, the attacker faces financial risk from paying for blocks on the attack chain only for the attack to fails.

The analysis in Section 3 assumes that block rewards are fixed and there is no difficulty adjustment. Gans and Halaburda (2024) show that an attacker can gain by affecting block rewards and, in particular, transaction fees. The attacker is advantaged as it can observe all the transactions processed by honest miners, and can collect all fees from transactions that are not invalidated by the attack.[74]

The difficulty adjustment can also be exploited by the attacker. For example, mining difficulty increases over time as mining equipment becomes more efficient, and the attacker can benefit from forking old blocks with lower difficulty using current mining equipment.

Centralized factors An attacker may be deterred by the threat of legal prosecution. 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.[75]

The analysis in Section 3.4 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 miner, 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 and to fail otherwise. To bribe such a big miner, it is no longer sufficient for the attacker to offer only a slightly higher than the normal block reward; it must also compensate for the individual cost to the big miner should the

[74] This may pose a limitation on an attacker that attempts to fork a very old transaction, as the fork will invalidate all the new coins given to miners in the forked blocks and all subsequent transactions these funds are exchanged in.

[75] There are many examples of hackers returning illicitly obtained funds; for example: https://www.reuters.com/technology/defi-platform-poly-network-reports-hacking-loses-estimated-600-million-2021-08- accessed August 2024.

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 economic security in this respect.

We point that while these factors may be effective in practice, they call into question the claims of the system's "decentralization".[76]

Discussion It is commonly believed that higher mining rewards and more mining power increase the economic security of a cryptocurrency. However, our economic analysis reveals interesting subtleties challenging this claim. Both of the economic environments we analyze suggest that higher mining rewards do not actually increase the cost of the attack. Overall, the connection between economic security and mining rewards may be more subtle or dependent on specific circumstances.

[76] Huberman et al. (2021) find that Bitcoin offers users protection from monopoly's harm even if there are large miners, as long as free entry of miners is possible. It is technically possible for a system to have both large pivotal miners for security and free entry of miners. However, in Nakamoto a sufficiently large miner is able to block free entry of miners.