Manifoldchain: Maximizing Blockchain Throughput via Bandwidth-Clustered Sharding

Chunjiang Che*, Songze Li†‡, Xuechao Wang*,
* The Hong Kong University of Science and Technology (Guangzhou)
†Southeast University
‡ Engineering Research Center of Blockchain Application, Supervision and Management (Southeast University), Ministry of Education

Abstract*—Bandwidth limitation is the major bottleneck that hinders scaling throughput of proof-of-work blockchains. To guarantee security, the mining rate of the blockchain is determined by the miners with the lowest bandwidth, resulting in an inefficient bandwidth utilization among fast miners. We propose Manifoldchain, an innovative blockchain sharding protocol that alleviates the impact of slow miners to maximize blockchain throughput. Manifoldchain utilizes a bandwidth-clustered shard formation mechanism that groups miners with similar bandwidths into the same shard. Consequently, this approach enables us to set an optimal mining rate for each shard based on its bandwidth, effectively reducing the waiting time caused by slow miners. Nevertheless, the adversary could corrupt miners with similar bandwidths, thereby concentrating hashing power and potentially creating an adversarial majority within a single shard. To counter this adversarial strategy, we introduce *sharing mining*, allowing the honest mining power of the entire network to participate in the secure ledger formation of each shard, thereby achieving the same level of security as an unsharded blockchain. Additionally, we introduce an asynchronous atomic commitment mechanism to ensure transaction atomicity across shards with various mining rates. Our theoretical analysis demonstrates that Manifoldchain scales linearly in throughput with the increase in shard numbers and inversely with network delay in each shard. We implement a full system prototype of Manifoldchain, comprehensively evaluated on both simulated and real-world testbeds. These experiments validate its vertical scalability with network bandwidth and horizontal scalability with network size, achieving a substantial improvement of 186% in throughput over baseline sharding protocols, for scenarios where bandwidths of miners range from 5Mbps to 60Mbps.

I. INTRODUCTION

Blockchain technology, pioneered by Nakamoto's proof-of-work (PoW) longest-chain protocol [1], has garnered substantial attention in recent years. PoW blockchains are distinct in offering unique properties like dynamic availability, unpredictability, and security against adaptive adversaries — merits that have been validated both theoretically [2], [3] and practically [4]. However, a fundamental challenge remains its inherent poor scalability, hindering its broader adoption and real-world applicability. A major impediment to scale PoW blockchains lies in the necessity for every miner to replicate the communication, storage, and state representation of the entire ledger. Blockchain sharding protocols have emerged as a promising solution to address this challenge. Sharding protocols allocate miners to distinct shards, where they independently mine separate chains within their respective shards [5], [6], [7], [8], [9], [10]. This innovative approach enables a linear scaling of system throughput in proportion to the network size.

However, a noteworthy shortcoming of most existing sharding protocols is the lack of attention to the bandwidth heterogeneity. Bandwidth variance can be substantial in real-world scenarios. For example, some Bitcoin miners operate within the Tor [11] network to safeguard their privacy, where only half of the available bandwidth in the Tor network is utilized [12], [13], [14], resulting in significantly smaller bandwidth compared to normal miners. Miners with limited bandwidth resources, often referred to as *stragglers*, contribute to increased network delays, which result in forking of blockchains, i.e., two distinct blocks extend the same preceding block. Forking significantly undermines blockchain security by diminishing the effective hashing power of honest miners, and elevating the probability of adversaries successfully tampering with ledger information. Consequently, fast miners are compelled to reduce their mining rates in order to mitigate forking, limiting overall transaction throughput. Most of the existing sharding protocols [5], [6], [8], [15], [16], [17], [7] typically employ a uniform shard formation (USF) mechanism across shards, resulting in each shard accommodating both stragglers and fast miners. Therefore, the throughput in each shard is still limited by bandwidths of stragglers.

We introduce *Manifoldchain* as a comprehensive solution to tackle this challenge. The key insight behind Manifoldchain is to cluster miners with similar bandwidths into the same shard, thereby segregating fast miners from stragglers. Consequently, an optimal mining rate can be set for each shard, tailored to bandwidth resources of miners within. This approach empowers fast miners to fully harness their computation resources, enabling them to propose blocks at a much higher rate without being impeded by stragglers. Manifoldchain employs a bandwidth-clustered shard formation (BCSF) mechanism. Specifically, miners package their band-

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.230028
www.ndss-symposium.org

width information, public key, and IP address as a "Pseudo-identity" (PID), and extend a credential chain with transactions encapsulating these PIDs. PID owners confirmed on the credential chain are subsequently distributed across distinct shards corresponding to their bandwidths. Once distributed, they begin processing transactions and growing ledgers within their respective shards.

A pivotal challenge in realizing such a sharding protocol is ensuring security within each shard in the presence of a mildly adaptive adversary (a general adversary model adopted by most sharding protocols). It may seek to corrupt miners with similar bandwidths, thereby concentrating adversarial hashing power within a single shard and potentially yielding an adversarial ratio exceeding 50%. To tackle this challenge, we introduce *sharing mining*, which utilizes cryptographic sortition to determine whether a miner will produce an *exclusive block* to a specific shard, or an *inclusive block* that can be appended to chains in all shards. Sharing mining enables Manifoldchain to aggregate the honest hashing power from the entire network to secure each individual shard, while simultaneously benefiting from boosted throughput due to various mining rates across those shards.

Nevertheless, as our design avoids full blocks in one shard from being sent to miners in other shards, it may lead to data availability and validity issues. We utilize *coded Merkle tree* (CMT) [18] and *fraud proof* [19] to verify data availability/validity without downloading the full blocks. Additionally, sharing mining incorporates strategies like *predictive mining* and *fork pruning* to parallelize mining and verification processes. Particularly, predictive mining empowers miners to mine on multiple unverified parent blocks while concurrently requesting data availability and validity proofs. Subsequently, miners can prune forks upon receiving unavailability or invalidity proofs.

Another challenge arises from the various mining rates across different shards when processing cross-shard transactions (*cross-txs*), which may lead to inconsistent cross-shard asset transfers. The standard atomic commitment protocol *Two-Phase Commit* (2PC) [20] employed by existing sharding protocols such as OmniLedger [6], is unsuitable for Manifoldchain. For instance, 2PC requires the verification process of a cross-tx to pause until confirmed vote messages are received from all involved shards, introducing prohibitive latency for confirming cross-txs. To address this challenge, we adopt an asynchronous commitment mechanism by eliminating the locking phase of 2PC. Specifically, coins are directly spent instead of being locked. Upon any unsuccessful expenditure, the coins are refunded to the payers.

We have conducted a comprehensive theoretical analysis of Manifoldchain, emphasizing its security and overall system throughput. Our analysis rigorously establishes the persistence and liveness properties of Manifoldchain, offering a precise upper bound on the error probability, a significant advance over the asymptotic results commonly found in previous works [21], [22]. This precise security characterization plays a crucial role in guiding the optimal configuration of protocol parameters. Notably, we address the determination of mining rates across different shards through an optimization problem, delivering an optimal solution for these settings. Finally, we formally prove that the optimal configuration of mining rates allows Manifoldchain to achieve a linear horizontal scaling with the number of miners, and a linear vertical scaling with the reciprocal of network delay in each shard.

Furthermore, to manifest the theoretical promise of Manifoldchain, we have implemented it concisely in about 15,000 lines of Rust code (available open source [23]). Our implementation has undergone comprehensive evaluation across diverse scenarios, encompassing a realistic testbed hosted on Amazon EC2, as well as a simulated testbed on a local machine. The experimental results from both realistic and simulated testbeds demonstrate that Manifoldchain outperforms baseline sharding protocols in several key aspects: (1) Manifoldchain delivers approximately $5\times$ throughput in the fastest shard configured with the highest mining rate, and approximately $3\times$ average throughput; (2) Manifoldchain achieves around a $3\times$ increase in throughput with the same increase in the number of miners; (3) Manifoldchain significantly boosts throughput with increased bandwidth resources, while baseline sharding protocols show no improvement.

II. Related Work

Vertically scaling protocols. The majority of research efforts focused on scaling blockchain performance vertically, namely designing consensus protocols with inherently high performance. Bitcoin-NG [24] achieves high throughput by decoupling Bitcoin's execution into two distinct primitives: leader election and transaction proposal. It decouples a full block into two parts: a key block for leader election and a microblock that contains transactions. The protocol relies on only key blocks with significantly reduced block size to expedite leader election, while allowing the elected leader to generate microblocks as fast as possible according to its computing/network resource. Similarly, Prism [25] embraces the same decoupling principle, allowing for concurrent leader election and transaction proposal processes. However, these protocols maintain a single ledger, making it difficult to scale with the network size.

Horizontally scaling protocols. To overcome the constraint encountered by vertically scaling protocols, several protocols aim to improve performance through horizontal scaling, which involves adding more nodes to distribute the load more effectively, thus boosting the system's throughput. Sharding stands out as the principal method for achieving Horizontal scaling. It divides the blockchain nodes into shards, each of which is only responsible for processing a distinct subset of the transactions. As a result, sharding enables the overall throughput to scale with the number of nodes.

Numerous works are dedicated to designing general sharding protocols in the permissionless setting. Elastico [5] uniformly partitions the mining network into small committees, within which miners run pBFT [26] to reach consensus on a disjoint set of transactions. Building upon this approach, Omniledger [6] addresses Elastico's limitations of not supporting cross-tx atomicity, by introducing a cross-tx verification mechanism called Atomix, which enables miners to verify cross-txs without storing the full blocks from foreign shards. RapidChain [8] further enhances the fault tolerance threshold from up to 1/4 in Omniledger to 1/3, boosts throughput within each shard through block pipelining, and reduces communication overhead via efficient routing. Nonetheless, these pBFT-based sharding protocols cannot achieve *full sharding* (as formally defined in Definition 1). Specifically, the aforementioned sharding protocols necessitate that the number of honest miners in each shard scales with the shard size. This scaling demand requires a significantly larger shard size to ensure an honest (super)majority within each shard, in accordance with the law of large numbers. In contrast, a full sharding protocol requires only a constant number of honest miners within each shard to ensure security. Wang *et al.* introduced Monoxide [7], the first full sharding protocol, which employs Chu-ko-nu mining to make attacking any specific shard as difficult as targeting the entire network. In the best-case scenario, this design can ensure security as long as each shard contains at least one honest miner. Nevertheless, Monoxide has its own shortcomings: (1) rigorous security analysis is not provided; (2) its proposed cross-tx verification mechanism, eventual atomicity, cannot support many-to-many transaction models.

Some other sharding solutions aim to tackle sub-problems within a sharding system. GearBox [15] and Reticulum[16] aims to attain the smallest shard size, thereby maximizing parallelism. Haechi[17] focuses on fortifying resilience against front-running attacks, wherein adversaries manipulate transaction execution order to achieve unfair finalization. ByShard[27] extends the system-specific specialized sharding protocols to a application-agnostic solution. All these works address various issues distinct from ours, yet Manifoldchain, as a general sharding solution, is compatible with them to address corresponding challenges.

Bandwidth considered protocols. It has been theoretically demonstrated that the network delay, denoted by Δ, plays an important role in determining an appropriate mining difficulty in synchronous blockchain protocols [28], [29], [30], [31]. Specifically, the block generation rate is constrained by Δ, as arbitrarily accelerating block generation can lead to excessive forking, thereby wasting honest mining power and compromising the network's security. Bitcoin-NG and Prism fundamentally enhance vertical scalability by mitigating the impact of Δ via functionality decoupling. However, they neglect the heterogeneity of blockchain nodes. Specifically, stragglers contribute to higher Δ values and subsequently become the bottlenecks for system throughput. To mitigate this issue, Yang *et al.* proposed DispersedLedger [32], a partial synchronous BFT protocol designed to achieve near-optimal throughput even in scenarios with heterogeneous network bandwidths. DispersedLedger enables nodes to agree on proposals at a rapid rate without downloading the entire block. Instead, nodes are able to retrieve the serialized transactions at their own paces. For instance, DispersedLedger nodes agree on Verifiable Information Dispersal blocks. These blocks utilizes erasure codes to store transactions across N nodes, ensuring that they can be retrieved later in the presence of Byzantine behavior. However, these non-sharding solutions inherently lack horizontal scalability.

Our protocol. We propose Manifoldchain, a permissionless full sharding protocol with salient attributes as follows:

- Manifoldchain is the first sharding protocol that takes the bandwidth heterogeneity into account. Other aforementioned protocols, either overlook the bandwidth heterogeneity in the sharding setup[5], [6], [8], [15], [16], [17], or only consider them in the non-sharding setup[24], [32].
- Manifoldchain achieves the highest fault tolerance threshold. Compared with pBFT-based sharding protocols[5], [6], [8], [15], [16], [17] whose fault tolerance threshold is up to 1/3, Manifoldchain supports a fault tolerance of up to 1/2 and achieves full sharding as Monoxide. Furthermore, it only requires that there is at least one honest miner in each shard. Compared with Monoxide, it addresses remaining limitation by providing a tight security analysis that guides the selection and optimization of protocol parameters. Additionally, Manifoldchain can support many-to-many cross-tx verification. Table I provides an overview of Manifoldchain in comparison to other sharding protocols.
- Manifoldchain is a general sharding principle that is compatible with many other sharding solutions. For instance, while GearBox and Reticulum improve horizontal scalability by maximizing number of shards, Manifoldchain offers a new angle to boost throughput within individual shards by allowing normal miners and stragglers to operate independently. Insights from previous works can be combined with ours to further improve both horizontal and vertical scalability.

III. Background and Model

A. Nakamoto Consensus Protocol

Nakamoto consensus (NC), adopted by Bitcoin, operates on the PoW mechanism and the longest chain rule. It can be described succinctly as follows: at any moment, an honest miner adopts the longest chain available to it and aims to mine a new block extending this chain; a block is considered confirmed once it is sufficiently deep within the chain.

Proof of Work. Finding a valid PoW solution necessitates locating a `nonce` value to ensures the output of the SHA256 hash function is less than the preset mining difficulty σ. We adopt the standard random oracle model [21], utilized by an algorithm $\mathbf{PoW}^{\sigma}(\texttt{parent\_hash}, \texttt{info}, \texttt{nonce})$ for finding a valid PoW solution. Here `parent_hash` and `info` are the hash of the parent block and block content respectively. The blockchain's immutability stems from the fact that any attempt to modify a single block within the chain inevitably triggers alterations to its hash value as well as to the hash values of all subsequent blocks.

TABLE I
COMPARISON OF MANIFOLDCHAIN WITH SOTA SHARDING PROTOCOLS

	Elastico [5]	Omniledger [6]	RapidChain [8]	Gearbox [15]	Reticulum [16]	Monoxide [7]	**Manifoldchain**
Global honest threshold[1]	$\frac{3}{4}N$	$\frac{3}{4}N$	$\frac{2}{3}N$	$\frac{2}{3}N$	$\frac{2}{3}N$	$\frac{1}{2}N$	$\mathbf{\frac{1}{2}N}$
Intra-shard honest threshold[2]	$\frac{2}{3}\frac{N}{m}$	$\frac{2}{3}\frac{N}{m}$	$\frac{2}{3}\frac{N}{m}$	$\frac{2}{3}\frac{N}{m}$	$\frac{2}{3}\frac{N}{m}$	≥ 1[3]	**1**
Supports many-to-many tx?	Yes	Yes	Yes	Yes	Yes	No	**Yes**
Supports full sharding?	No	No	No	No	No	Yes	**Yes**
Scalable with bandwidth?	No	No	No	No	No	No	**Yes**

[1] The minimum number of honest miners necessary to ensure the security of the entire network, where N is the total number of miners.
[2] The minimum number of honest miners necessary to ensure the security of a specific shard, where m is the number of shards.
[3] In best-case scenario, Monoxide can achieve the same intra-shard honest threshold as Manifoldchain, but it lacks a formal security analysis.

Unspent Transaction Output (UTXO). A Bitcoin transaction consists of multiple inputs and outputs, each represented as an UTXO. For a Bitcoin transaction $\texttt{tx} : \{\mathrm{I}_1, ..., \mathrm{I}_\mathrm{k}; \mathrm{O}_\mathrm{p}, ..., \mathrm{O}_\mathrm{h}\}$ with $\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathcal{A}(\mathrm{I}_\mathrm{k}) = \sum_{\mathrm{j}=1}^{\mathrm{h}} \mathcal{A}(\mathrm{O}_\mathrm{j})$(where $\mathcal{A}(\cdot)$ represents the amount of coins), k inputs are consumed and h outputs are generated. Transaction outputs may function as inputs for subsequent transactions; however, an output can be utilized only once. In other words, UTXOs embody the available outputs that can be employed to generate new transactions, effectively representing the user's balance in Bitcoin network.

B. Blockchain Sharding Protocol

A blockchain sharding protocol partitions miners into different shards, each processing a disjoint set of transactions. Overall, a specific sharding protocol involves two phases: shard formation phase and ledger generation phase. In the shard formation phase, sharding protocols utilize a shard formation mechanism to partition miners into distinct shards. A standard approach is the USF mechanism, which randomly shuffles miners and uniformly distributes them into shards. This initial design ensures that adversarial hashing power is evenly distributed among multiple shards, thereby fortifying each shard against targeted attacks. In the ledger generation phase, an intra-shard consensus protocol like NC or pBFT is employed by miners to reach agreement on the transaction order and update the ledger state within each shard. The choice between NC and pBFT is scenario-dependent: pBFT is preferred for its faster transaction confirmation times, whereas NC is favored for its adaptability to variable shard sizes. Additionally, cross-shard atomicity must be ensured to guarantee the atomic transfer of assets across shards. Specifically, the outcome of a cross-shard transaction—whether committed or aborted—is consistent across all involved shards.

Full Sharding Protocols. Let N represent the number of miners and m represent the number of shards, we define a full sharding protocol as follows:

Definition 1. *A sharding protocol achieves full sharding if there exists a constant n_0, such that for all m and N satisfying $0 < m \leq N$, security holds as long as each shard contains at least n_0 honest miners.*

Intuitively, a full sharding protocol requires only a constant number of honest miners in each shard to ensure security. Early sharding protocols, such as Elastico [5], Omniledger [6], and Rapidchain [8], require a linearly scaling number of honest miners in each shard (generally over two-thirds of the miners) and are categorized as non-full sharding protocols. Wang *et al.* introduced the first full sharding protocol Monoxide [7]. In the best-case scenario, it requires only one honest miner in each shard. Our proposed protocol, Manifoldchain, is a more complete full sharding protocol equipped with formal security analysis and a many-to-many transaction design.

C. Network Model

Given that NC-style protocols are known to lack security in a partial synchronous network [33], [28] where there is an unknown bound on the network delays, we adopt the standard synchronous model. This model constrains the adversary to delaying messages from honest nodes by no more than a known maximum delay, denoted as Δ. Previous works [21], [33], [28], [22], [3] have consistently overlooked the network heterogeneity, instead assuming the worst-case end-to-end delay to be Δ. This simplification leads to an elegant trade-off between security and throughput in NC, as shown by Pass et al. [31]. Let $\rho \in (\frac{1}{2}, 1]$ be the fraction of honest hashing power and N the total number of miners. Pass et al. have theoretically established that to prevent consistency violations, the difficulty parameter p must be set below $\frac{1}{(1-\rho)N\Delta}$, where $p = \sigma/2^{256}$ is the probability of a successful outcome from a single query to the random oracle $\mathbf{PoW}^{\sigma}(\cdot)$. Consequently, as the difficulty parameter p determines the block generation rate, the throughput is directly proportional to p and inversely affected by an increase in delay Δ.

In this work, we adopt a standard network model adopted by many other works [24], [25], [32], [34], [35]. Specifically, miners have various bandwidths, represented as $\mathtt{C}_i$ for miner i. The network delay consists of two components: transmission delay Δ_t and propagation delay Δ_p. Transmission delay refers to the duration required to push all bits of a message into the link, calculated as $\Delta_t = \frac{m}{\mathtt{C}}$ for a message of m bits and a link bandwidth of $\mathtt{C}$ bits per second. Propagation delay Δ_p, on the other hand, measures the time it takes for a bit to travel from one end of the link to the other, dependent on the distance and the speed of light or electrical signal propagation. Consequently, let $\Delta_{\mathtt{C}} = \mathtt{B}/\mathtt{C} + \Delta_p$ be the network delay for a node with bandwidth $\mathtt{C}$, where $\mathtt{B}$ is the maximum block size. We denote by $\Delta = \max_i \Delta_{\mathtt{C}_i}$ the maximum delay in the network, primarily determined by the straggler with the smallest bandwidth.

D. Threat Model

We consider both static and dynamic settings in this paper, tailored to address distinct types of adversaries. In the static setting, we consider a *static* adversary [36] who corrupts a fixed set of miners when the protocol starts. In contrast, an *adaptive* adversary can change which miners to corrupt during protocol execution. In the dynamic setting, we consider a *mildly* adaptive adversary[6], which requires a certain delay to transfer corruption. Other types of adaptive adversaries, including weakly and strongly adaptive adversaries [37], [38], [39], lack this delay and can corrupt all miners in a shard at any time, are not considered in typical sharding protocol designs [6], [8], [5], [40]. A detailed comparison of the mildly adaptive adversary with other adaptive adversaries is provided in Appendix A. We present the detailed adversary model as follows:

- In the static setting, the set of participating miners, totaling N, remains unchanged with no new entries or exists during the protocol's operation. This scenario assumes a static adversary who, prior to the protocol's start, can corrupt up to $(1-\rho)N$ miners. The adversary does not have prior knowledge about the miners' participation or specific attributes (particularly their bandwidth information). Once the protocol begins, the adversary cannot modify the selection of corrupted miners.
- In the dynamic setting, miners can join or leave the network during the shard formation phase, but their participation stabilizes during the ledger generation phase. Formally, we denote by α_i the fraction of the new miners among all miners in the i-th shard formation phase. Specifically, in the initialization phase and each shard formation phase, the adversary can corrupt up to $1-\rho$ fraction of participating miners or new miners without prior knowledge of their bandwidth specifics. Additionally, the adversary has the capability to dynamically change which miners to corrupt but any transition to corrupt additional miners demands a minimum time investment equivalent to the duration of each ledger generation phase. Regardless of the adversary's corruption strategy, the maximum fraction of corrupted miners remains $1-\rho$.

Both static and adaptive adversary can arbitrarily determine the participating strategy of the corrupted miners, and fully control their associated communication and computation resources. We assume there exists an underlying bandwidth distribution of honest miners, and the new honest miners joining in each shard formation phase adhere to this distribution. Prior to protocol execution, we conduct bandwidth estimation, which can be achieved by leveraging existing techniques [41], [42], even under Byzantine settings [43].

IV. Manifoldchain

Overview. We introduce Manifoldchain, the first blockchain sharding protocol that simultaneously scales system throughput vertically and horizontally. The key idea is using the BCSF mechanism to allocate miners with similar bandwidths to the same shard. This mechanism clusters stragglers into some specific shards, mitigating their impact on the performance of other shards. As a result, we can tailor the mining rate for each shard based on its specific network delay, and fast miners can create blocks and process transactions at an accelerated pace, leading to an overall increase in throughput. Fig. 1 demonstrates our key insight. Here, we emphasize two primary challenges that are intrinsic to the design of such a blockchain sharding protocol.

The first challenge is to maintain security within each shard. The corrupted miners may misstate similar bandwidths, concentrating adversarial hashing power in a single shard, resulting in an adversarial majority in that shard. We propose sharing mining to ensure the security within those shards with an adversarial majority. It is implemented by allowing miners to simultaneously extend multiple longest chains in all shards using a single PoW solution, which enables honest miners to contribute their hashing power across all shards. However, in sharing mining, only the block header - rather than the full block - is sent to miners in other shards. The adversary can potentially create equivocation on the highest block through sending heads of invalid blocks to other shards, splitting the honest hashing power within and outside of a shard. We address this issue by employing predictive mining and fork pruning. Predictive mining enables miners to mine on multiple potential parents and try to request data availability and validity proof at the same time, while fork pruning serves to prune forks with invalid availability or validity proofs.

Another challenge is to develop an asynchronous atomic commitment mechanism to ensure the atomic multi-input-multi-output UTXO transfer across shards under various mining rates. The standard atomic commitment 2PC [20] incorporates Voting Phase and Decision Phase to handle this problem. Specifically, a coordinator node prompts participants to vote, and if all vote to commit, broadcast a commit message; otherwise, broadcast an abort message if any votes to abort. In Manifoldchain, the coordinator node cannot begin Decision Phase until all participants' vote messages are confirmed. The participants may be stragglers from slows shards, and their long confirmation time results in an unacceptable cross-tx latency. To address this challenge, we implement an asynchronous commitment mechanism by eliminating the locking phase. Initially, coins are expended directly rather than being locked if input shards vote to commit. Upon receiving all messages voting to commit, the corresponding coins will be subsequently generated in the output shards. Conversely, if any shard votes to abort or any vote message becomes outdated, the associated coins will not be generated, and the coins will be refunded to the input shards with prior successful expenditure.

A. Bandwidth-Clustered Shard Formation

We propose the BCSF protocol such that miners with similar bandwidths are allocated into the same shard. In BCSF, all miners maintain a *credential-chain* based on NC, incorporating

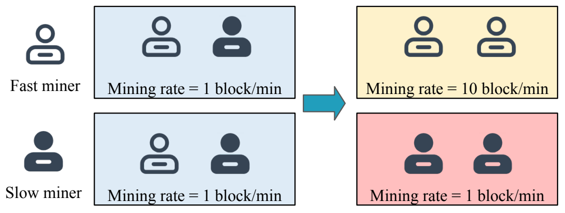
Fast miner
Mining rate = 1 block/min
Mining rate = 10 block/min
Slow miner
Mining rate = 1 block/min
Mining rate = 1 block/min

Fig. 1. Basic insight of Manifoldchain. The hindrance imposed by bandwidth heterogeneity is alleviated by gathering the stragglers.

miner PID in place of transactions into the chain. A PID contains a miner's public key, the signed bandwidth using its secret key, and his IP address. The mining process of the credential-chain mirrors that of NC: miners package identities into blocks and strive to find a valid PoW solution. Upon receiving a block, each miner undertakes two verification processes: the first verifies the correctness of the PoW solution, while the second verifies the signature of the bandwidth using the public key provided in the identity. Given the liveness parameter u of the credential-chain, which determines the maximum time it takes for a transaction to be confirmed, and the start time t of the shard formation phase, we ensure that all PIDs of honest miners are confirmed by $t + u$. The owner of a confirmed PID within the time interval $[t, t + u)$ acquires the authorization to participate in the ledger generation phase.

We represent the information of these authorized miners in a two-dimensional space. The domain of the X-axis spans from 0 to 2^{256}, while the Y-axis is defined within the range of $(\mathtt{C}_{min}, \mathtt{C}_{max})$, where $\mathtt{C}_{min}$ and $\mathtt{C}_{max}$ represent the minimum and maximum bandwidth in the network respectively. The coordinate of a miner, represented as $(h, \mathtt{C})$, is composed of the hash value of its PID h and bandwidth $\mathtt{C}$. Subsequently, different shard formation mechanisms specify how to partition the space and distribute miners into different shards.

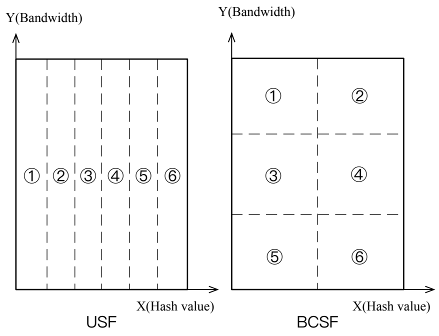

Fig. 2. USF vs BCSF. The first one uniformly distributes miners across shards while the second one clusters miners with similar bandwidths.

Shard formation mechanism. We abstract the USF mechanism as follows: the space is divided into m regions along the X-axis, each corresponding to one shard. Given that hash values can be treated as random numbers, the USF mechanism effectively ensures a uniform distribution of miners across shards; BCSF incorporates an additional consideration of nodes' bandwidth information: the space is partitioned into multiple regions along both the X-axis and Y-axis, facilitating the grouping of miners with similar bandwidths. An example for $m = 6$ shards is shown in Fig. 2.

Secure shard partition. We denote by S_X and S_Y the numbers of segmented regions along X-axis and Y-axis, respectively. Increasing both S_X and S_Y improves overall throughput by generating more shards. However, it's crucial not to overly expand the number of shards, as doing so may result in shards devoid of honest miners, especially if the total shard count surpasses the number of honest participants. Let $R_Y(\underline{y}, \overline{y})$ represent a Y-region containing points with Y-coordinates within the range $(\underline{y}, \overline{y}]$. A shard formation mechanism segments the space into S_Y Y-regions.

We can set the Y-region ranges to achieve a uniform distribution of miners across these regions. Particularly, given an estimated bandwidth distribution, we select $S_Y - 1$ separation points $y_1, \ldots, y_{S_Y-1}$, such that

$$P(y_0, y_1) = P(y_1, y_2) = \cdots = P(y_{S_Y-1}, y_{S_Y}), \tag{1}$$

where $y_0 = \mathtt{C}_{min}$, $y_{S_Y} = \mathtt{C}_{max}$, and $P(a, b)$ denotes the probability that a miner's bandwidth is within $(a, b]$.

During the ledger generation phase, miners with the authorization sign the generated blocks using their secret keys and subsequently broadcast these blocks to other miners. The recipients validate the incoming blocks using the public keys stored within the confirmed PIDs. For rotation, miners update bandwidth information by broadcasting new PIDs during each shard formation phase. They then access updated shard allocation information from the credential chain and dynamically re-shard the network.

Selection of S_X and S_Y. We demonstrate how to select appropriate values for S_X and S_Y to ensure that each shard contains at least one honest miner, as illustrated in Lemma 1. Initially, we set S_Y as large as possible to effectively separate fast and slow miners, while minimizing bandwidth variance within each Y-region. Given a fixed S_Y, we set S_X as large as possible to increase the number of shards, as long as the error probability in (2) is negligible. For instance, if all miners have either high or low bandwidth, we set $S_Y = 2$ and increase S_X until the error probability exceeds a predefined threshold.

Adversary-proofness of BCSF. Two malicious deviations can occur during the shard formation phase. Firstly, a malicious miner might attempt to enter a specific shard by manipulating its PID, both the hash value and bandwidth information, submitted to the credential-chain. However, by carefully choosing the values of S_X and S_Y, BCSF ensures the presence of at least one honest miner in every shard, irrespective of the adversary's choices of shards. This condition is sufficient to establish the security of Manifoldchain, as elaborated in Section V. At a high level, the security and performance of each shard remain intact despite the presence of malicious miners with incompatible bandwidths, thanks to the Byzantine fault tolerance inherent in our protocol. Secondly, a malicious miner may submit multiple PIDs to the credential-chain. However, during the ledger generation phase, such a miner must either distribute its hashing power among various PIDs or concentrate it on a single one. Either case does not change the aggregate adversarial mining power across shards, rendering the strategy ineffective. Note that we rely on the liveness property of NC to ensure that all honest miners' PIDs are confirmed on the credential-chain before the shard formation phase concludes. Attacks at the network layer, such as spamming, fall outside this paper's scope.

B. *Block & Chain Structures*

Our block structure decouples the traditional Bitcoin full block into two types of blocks: the *consensus block* and the *transaction block*. A consensus block has a fixed size of approximately 100 bytes, introducing negligible overhead for both communication and storage. Consensus blocks are broadcast to all miners in the network, while transaction blocks are only transmitted within their own shards. To support the implementation of sharing mining, we categorize consensus blocks into exclusive blocks and inclusive blocks.

Distinct from a Bitcoin full block, a consensus block could have multiple parents. More specifically, an exclusive block extends chains within its affiliated shard and can have multiple parents, whereas an inclusive block, with the ability to have parents from multiple shards, extends chains of all shards. This design, allowing for multiple parents, facilitates predictive mining where miners may have to mine on multiple unverified blocks. This will be further elaborated in Section IV-C. We employ the idea of 2-for-1 PoW [21] to mine both exclusive and inclusive blocks concurrently. Initially, miners engage in the process of mining a consensus block and they do not know the type of the consensus block until the puzzle is solved. When a consensus block is mined, it is considered mined of either exclusive block or inclusive block depending on the region the block hash falls. Specifically, A miner mines a consensus block when it discovers a nonce that satisfies $\mathbf{PoW}^{\sigma}(\cdot, \mathtt{nonce}) < \sigma$. Further, we can set a threshold $\sigma' < \sigma$. If $\mathbf{PoW}^{\sigma}(\cdot, \mathtt{nonce}) < \sigma'$, the consensus block is deemed an inclusive block; otherwise, it is classified as an exclusive block. We present all the segments of a consensus block as follows:

- `shard_index`: an index denoting the specific shard to which the block is affiliated.
- `verified_parent`: hash of the highest verified block within the affiliated shard.
- `inter_parents`: a set composed of all unverified blocks' hashes extending the highest verified block within the affiliated shard.
- `global_parents`: a set composed of all inter parents across all shards.
- `timestamp`: the time when the block is generated.
- `nonce`: the resolved solution to a PoW puzzle within the context of sharing mining.
- `tx_merkle_root`: the root of a Merkle tree generated from all transactions in a transaction block.
- `tmy_merkle_root`: the root of a Merkle tree generated from all testimonies in a transaction block.

Adopting the UTXO model, a transaction in Manifoldchain consist of multiple inputs and outputs. Each input contains a data field called `payer_addr`, signifying the address of the payer. Similarly, each output contains a data field named `receiver_addr`, denoting the receiver's address. Basically, transactions are categorized into two main types: domestic transactions (domestic-txs) and cross-txs. If all payers and receivers involved in a transaction belong to the same shard, the transaction is termed a domestic-tx; conversely, if participants span multiple shards, it is referred to as a cross-tx. A transaction block resembles transactions is referenced to a consensus block through the `tx_merkle_root`. Within a transaction block each cross-tx is accompanied by an additional testimony. This structure offers an efficient method for verifying cross-txs, presented in Section IV-D.

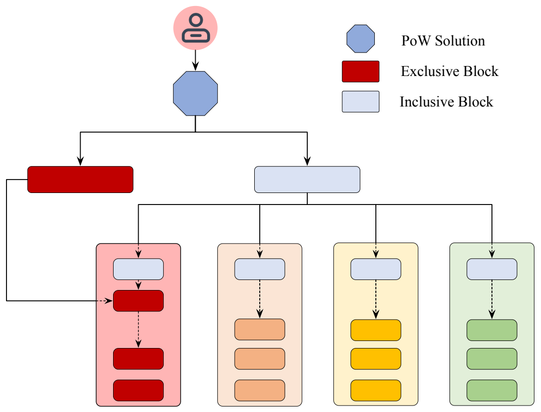

Fig. 3. Overview of sharing mining. An exclusive block extends the blockchain within a specific shard, whereas an inclusive block extends all blockchains across shards.

C. *Sharing Mining*

Using the BCSF shard formation mechanism, honest hashing power is evenly distributed across all shards, but adversary could concentrate its hashing power on specific shards. To ensure security in shards with adversarial majority, we propose sharing mining to allow honest miners to contribute their hashing power to all shards. An initial design of sharing mining, inspired by Chu-ko-nu mining, allows a miner to extend chains in all shards simultaneously using a single PoW solution. Instead of mining on one parent block, a miner works on creating a consensus block using the hash of highest blocks from all shards. Adopting 2-for-1 PoW, the consensus block is then classified as either an exclusive block or an inclusive block. In the case of an exclusive block, it extends the longest chain within the miner's corresponding shard. Conversely, in the case of an inclusive block, it simultaneously extends multiple longest chains across all shards. Honest miners treat exclusive blocks and inclusive blocks equivalently, adhering to the same verification and fork selection mechanisms. Fig. 3 shows how sharing mining works.

Hashing power splitting attack. As the transaction block associated with a consensus block is not sent to miners outside the shard, it causes the protocol vulnerable to a Hashing Power Splitting Attack (HPSA). Specifically, an attacker with overwhelming hashing power within a shard may mine a longer chain, containing invalid transactions but appearing valid from the perspective of honest miners in other shards, as they cannot verify the validity of the block without the transactions. This can cause honest miners to mine on different consensus blocks, leading to split of honest mining power and hence reduced security. A detailed description of HPSA is given in the Appendix B.

To counter HPSA, Manifoldchain requires each miner to verify the following properties of each block from foreign shards, without storing the corresponding transaction block.

- **Transaction validity** ensures that all transactions packaged within a block adhere to the appropriate format and preclude any double-spending events.
- **Data availability** requires a full block including both consensus block and associated transaction block, appears in each honest miner's view within its shard.

We provide corresponding verification mechanisms for these two properties in scenarios where there is at least one honest miner in each shard. To ensure transaction validity, we employ fraud proof [19] to enable in-shard miner to inform out-shard miner of a block's violation of validity rules. When an honest in-shard miner encounters an invalid block, it generates a fraud proof and broadcast it among all out-shard miners. Upon receiving a valid fraud proof, honest out-shard miners reject the block.

For data availability, we utilize CMT [18] to enable miners to verify data availability without downloading the full block. Specifically, CMT adds redundancy to transaction blocks via erasure codes. Any honest miner can verify full data availability through the mere download of a block hash commitment with a size of $\mathcal{O}(1)$ byte, combined with a random sampling of $\mathcal{O}(\log(B))$ bytes. Therefore, out-shard miners can validate the data availability of an in-shard transaction block by requesting just $\mathcal{O}(\log(B))$ samples. When an honest out-shard miner receives a foreign block, it requests block samples from the source shard. If it receives any error-coding proofs or fails to receive all requested samples within the maximum network delay Δ, it rejects the block. Detailed verification mechanisms are provided in Appendix C.

Verifying data availability and transaction validity can take up to the maximum network delay Δ to complete, diminishing the benefit of Manifoldchain by clustering miners with similar bandwidths. To maintain the throughput gain from downloading fraud and data availability proofs from foreign shards, we further optimize sharing mining, by proposing novel techniques of *predictive mining* and *fork pruning*. The key idea is for the miner to simultaneously mine on multiple blocks, with some of them waiting to be verified by messages from other shards. By doing this, a miner can start mining on a foreign block as soon as receiving the block, without needing to wait for verification. Upon completion of the verification process, a pruning procedure is conducted to maintain the uniqueness of the ledger. Consequently, any forks that do not extend the the longest verified chain are discarded.

Predictive mining. An exclusive block has multiple parents within one shard, comprising the last blocks of the longest verified chain and all unverified forks that extend on the

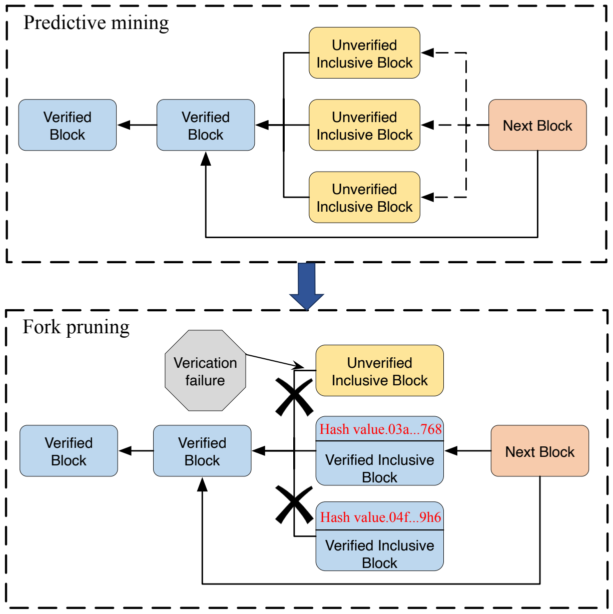

Fig. 4. Key insight of predictive mining and fork pruning. Blocks can extend potential parents and invalid forks are subsequently pruned.

longest verified chain. This exclusive block temporarily extends all chains that are possibly the longest verified chain. An inclusive block has multiple parents across all shards. Within each shard, similar to exclusive block, its parents comprise the last blocks of the longest verified chain and all unverified forks in that shard. As illustrated in Fig. 4, three unverified blocks extend from the second verified block. By employing predictive mining, the subsequent block will consider all three unverified blocks, along with the second verified block, as parents and extend its chain on top of them.

We present the procedures for mining exclusive and inclusive blocks, along with their verification process as follows:

- **Mining**: a miner gathers all requisite segments for generating a consensus block. Initially, it package valid transactions and testimonies into a transaction block and generate the corresponding `tx_merkle_root` and `tmy_merkle_root`. As for other segments, `verified_parent` is designated as the hash value of the last block of the longest verified chain; `inter_parents` is a vector containing `verified_parent` and the hashes of the last blocks of all unverified forks that are at least as long as the longest verified chain; The `global_parents` is formed through the aggregation of the `inter_parents` across all shards. The miner works on finding a solution for $\mathbf{PoW}^{\sigma}$(`parent_hash`, `info`,`nonce`). In this context, `parent_hash` refers to the hash of the three segments comprising `verified_parent`, `inter_parents`, and `global_parents`. All other segments, excluding `nonce`, undergo hashing to formulate `info`. The generated consensus block is subsequently categorized as an exclusive block or inclusive block based on its hash value, and it is linked to the transaction block. Both exclusive blocks and inclusive blocks are sent to all miners, while transactions blocks are only broadcast within their corresponding shards.

- **Verification**: when a miner receives an exclusive or inclusive block from other shards, it first checks the correctness of the PoW solution by recomputing the hash value. If the solution is correct, the miner examines each parent included in the block. If a parent is verified invalid or does not exist in all possible forks, the miner gives up extending the block on this parent. Otherwise, it extends the block on the parent, even if it is not part of the longest chain (storing an exclusive or inclusive block incurs negligible storage as it does not contain any transactions). Conversely, if a miner receives an exclusive or inclusive block belonging to its shard, in addition to the aforementioned verification procedure, it also verifies the validity of the linked transaction block. For each domestic transaction, it conducts standard verification akin to Bitcoin's process. For each cross-shard transaction, it follows the verification method outlined in Section IV-D. The miner rejects blocks containing any invalid transactions.

Fork pruning. Once an unverified block is successfully verified with both transaction validity and data availability, some forks are temporarily considered invalid and subsequently excluded from the parent blocks during the mining of the next block. Pruning is employed under the following two situations:

1) **Verification failure**. If a block fails the data availability verification (a miner has not received enough samples to complete the availability verification of a block within 2Δ seconds since receiving the block) or the transaction validity verification fails (a miner receives a valid `proof_of_invalidity` of that block within 2Δ seconds since receiving the block), any forks extending from this block are pruned.
2) **Sibling block with lower hash value is verified**. In cases where multiple blocks on the same level attain verification, the block possessing the lowest hash value stays, while its sibling blocks are subsequently pruned.

Forking rate about sharing mining. Even though sharing mining enables both exclusive blocks and inclusive blocks to extend on multiple parents, it does not produce any extra unexpected forks. (1) An exclusive block with multiple parents results in extra forks, but these forks are expected and only exist for a maximum interval of Δ seconds. After this interval, only one fork remains, indicating that if we solely consider the verified portion of a chain, our sharing mining protocol aligns with the insights of the NC protocol. (2) An inclusive block does not lead to any additional forks. Upon finding a PoW solution for an inclusive block, a miner generates both the inclusive block and its associated transaction blocks. The inclusive block solely contains essential header information and is negligibly small compared to the transaction block. Only the inclusive block is transmitted to other shards, while the transaction block remains within the shard. The transmission delay for inclusive blocks is negligible, thereby preventing the occurrence of extra forks when transmitting inclusive blocks across shards.

D. Cross-tx Atomicity

In the context of a sharding protocol, miners across different shards must collectively reach consensus on whether a cross-tx should be committed or not. This scenario aligns with the classic atomic commitment problem in distributed databases [44], [45]. As discussed in Section IV-A, transactions are either domestic-txs or cross-txs. A cross-tx involves multiple inputs and outputs originating from different shards. Here "input" signifies the expenditure of coins, while "output" denotes the creation of coins. We refer to shards in which coins are spent as "input shards", and shards in which coins are created as "output shards". The atomic commitment across shards requires that if all inputs are confirmed as spent, all outputs should be successfully created. Conversely, if any input is confirmed as invalid, all outputs should not be generated and other inputs are not spent eventually. The most straightforward and widely recognized approach for achieving atomic commitment is 2PC, a method employed by existing blockchain protocols such as RSCoin[46] and OmniLedger. Specifically, the coordinator node prompts participants to vote, and if all vote to commit, broadcast a commit message; otherwise, broadcast an abort message if any votes to abort.

New challenge arises from sharding setting with various mining rates when naively applying 2PC to Manifoldchain: The vote messages may become outdated when forking occurs. For instance, a shard initially votes to commit a cross-tx, but the inputs may become invalid upon a forking occurrence. An intuitive solution is to delay the decision phase until the confirmation of all vote messages, which is adopted by OmniLedger. However, this solution requires miners in fast shards to pause the decision phase before receiving all confirmed vote messages, resulting in unacceptable cross-tx latency and contradicting our initial motivation. To address this challenge, we implement an asynchronous commitment mechanism by eliminating the locking phase. Initially, coins are expended directly rather than being locked if input shards vote to commit. Upon the successful expenditure of all inputs, the corresponding coins will be subsequently generated in the output shards. Conversely, if any input shard votes to abort or any vote message becomes outdated, the associated coins will not be generated, and the coins will be refunded to the input shards with prior successful expenditure.

We introduce testimony to enable miners within the output shards to ascertain the successful expenditure of inputs. We implement the execution of a cross-tx by broadcasting it to each input shard and output shard. Cross-txs within the input shards are designated as input-txs, whereas those situated within the output shards are labeled as output-txs. Each output-tx is associated with a testimony to enable miners to verify the validity of its corresponding input-txs asynchronously. A testimony is composed of multiple units, each corresponding to an input. Essentially, each testimony unit maintains a Merkle proof for corresponding input-tx that proves its inclusion within the longest chain. Specifically, the structure of a testimony unit is presented as follows:

Input hash. The hash of the corresponding input.

Originate block hash. The hash of the originate block where the input-tx is packaged.

Transaction Merkle proof. A merkle proof which proves the inclusion of the input-tx in the originate block.

Vote message. An additional bit indicates an acceptance or rejection.

When a miner generates a new block, for each input-tx, it generates the corresponding testimony and sends it to all involved output shards. As the inputs of a cross-tx are distributed among different shards, each shard can create a partial testimony encompassing units for a portion of inputs. After receiving all partial testimonies, miners in the output shards combine them into one full testimony, composed of corresponding units for all inputs. The full protocol is illustrated as follows:

1) **Initialization**. A user creates a cross-tx and gossips it to all involved input shards and output shards.
2) **Expenditure**. When a miner in the input shards receive an input-tx, it proceeds as follows. First, it decides whether the coins can be spent based on the ledger information. If the input-tx is valid, the miner labels this input-tx as accepted and includes it in the mining pool. In the case of an invalid input-tx, it is labeled as rejected but is still placed in the mining pool. Upon successfully mining a valid block, for each involved accepted or rejected input-tx, the miner generates a testimony containing a vote message of corresponding acceptance or rejection and broadcasts it to all output shards. Aligning with its label, an input-tx is confirmed as either accepted or rejected upon reaching a κ-confirmation (the block containing the input-tx is followed by κ consensus blocks).
3) **Receipt**. When a miner in the output shards receives an output-tx along with all associated testimonies for all inputs, it is included in the mining pool. Output-txs are subsequently packaged into blocks and appear in the longest chain. For each output-tx in the longest chain, if all associated inputs are confirmed in other shards, miners validate their validity by verifying the corresponding testimonies. If all input-txs are confirmed as accepted, the output-tx is labeled as accepted. Conversely, if any input-txs are confirmed as rejected or testimonies are found to be invalid, the output-tx is labeled as rejected, triggering the subsequent refunding process. When an output-tx reaches a κ-confirmation, it is confirmed as either accepted or rejected in accordance with its label.
Refund. Upon the κ-confirmation of a rejected output-tx and all corresponding input-txs, miners generate a refund transaction (refund-tx) along with an *proof-of-rejection*. This proof-of-rejection contains two testimonies $\texttt{tes}_I$ and $\texttt{tes}_O$. $\texttt{tes}_O$ proves the rejected output-tx is included in the corresponding output shard's longest chain. $\texttt{tes}_I$ testify either the inclusion of a rejected input-tx in the longest chain of the corresponding input shard, or the inclusion of an input-tx in a deconfirmed block (where another block at the same height is confirmed) of the input shard. Subsequently, both the refund-tx and proof-of-rejection are broadcast to all input shards. Upon receiving a refund-tx and the proof-of-rejection from all output shards, miners in the input shards can verify the validity of the refund-tx through the proof-of-rejections. If all output-txs are confirmed as rejected, the refund-tx is labeled as accepted, miners include it in the blockchain to refund the coins to the input shards. Similarly, a refund-tx is confirmed as neither accepted or rejected based on its label upon a κ-confirmation.

This protocol ensures cross-tx atomicity across shards, and the detailed proof is provided in Appendix D. Intuitively, honest miners confirm an output-tx only if all input-txs are confirmed. If an adversary attempts to disrupt the atomicity of a cross-tx, they must target one of the input-txs and include a malicious input-tx with a different decision in a block. They must then ensure that this block is confirmed at the same level as the block containing the original input-tx. However, our sharing mining protocol ensures that, with overwhelming probability, no two different blocks at the same level are confirmed by honest miners, thereby preventing the adversary from breaking atomicity.

We point out that cross-tx latency depends on the slowest shard, as output-txs cannot be confirmed before all input-txs are confirmed. This bottleneck, shared by other sharding protocols, arises from the need for atomicity: decisions require votes from all participants. However, when cross-txs are not concentrated in the slowest shard, our protocol achieves lower average latency than baseline sharding protocols, as shown in Section VI-A.

V. Security and Scalability Analysis

A. Honest Presence

The security of Manifoldchain relies on the presence of at least one honest miner in each shard. Recall that α_i represents the fraction of new honest miners joining during the i-th shard formation phase, the following lemma demonstrates that Manifoldchain holds an honest presence within each shard.

Lemma 1. *Let $\underline{\alpha} = \min_i \alpha_i$, i.e., the minimum fraction of newly joined miners among all miners in a shard formation phase. Given parameters $1 \le S_Y \le \max\{\underline{\alpha}\rho N, \lceil \frac{1}{1-\rho} \rceil - 1\}$ and $S_X \ge 1$, our shard formation mechanism ensures that each shard contains at least one honest miner, except with a negligible error probability bounded by ε, where*

$$\varepsilon = \begin{cases} (S_X S_Y - 1)(\frac{S_X S_Y - 1}{S_X S_Y})^{\underline{\alpha}\rho N - 1} & S_Y \ge \frac{1}{1-\rho} \\ (S_X S_Y - 1)(\frac{S_X S_Y - 1}{S_X S_Y})^{\rho N - 1} & 0 < S_Y < \frac{1}{1-\rho} \end{cases} \quad (2)$$

The detailed proof of this lemma is available in Appendix G of the complete paper [47], where we also show that even under an imperfect bandwidth estimation with small deviation, honest presence still holds in each shard except with a negligible error probability for overwhelmingly large N.

We meticulously follow the selection strategy presented in Section IV-A to choose S_X and S_Y, ensuring that Lemma 1 holds–each shard contains at least one honest miner with high probability. Specifically, in a scenario with a sufficient number of new honest miners, i.e., $\underline{\alpha}\rho N > \lceil \frac{1}{1-\rho} \rceil - 1$, our shard partition mechanism ensures that each Y-region has approximately $\frac{\alpha\rho N}{S_Y}$ new honest miners, which are uniformly distributed across S_X shards. Therefore, each shard averages $\frac{\underline{\alpha}\rho N}{S_X S_Y}$ new honest miners, and Lemma 1 holds with high probability, by Chebyshev's inequality. Conversely, in a scenario with few new honest miners, i.e., $\underline{\alpha}\rho N \leq \lceil \frac{1}{1-\rho} \rceil - 1$, we set $S_Y = \lceil \frac{1}{1-\rho} \rceil - 1$ and uniformly distribute all honest miners in each Y-region across S_X shards. Similarly, each shard averages $\frac{\rho N}{S_X S_Y}$ honest miners, and Lemma 1 holds with high probability.

B. Consistency and Liveness

Next, we demonstrate that given honest presence in each shard, Manifoldchain attains three basic security properties: Common Prefix (CP), Chain Quality (CQ), and Chain Growth (CG), formally defined in Appendix E. The CP property implies consistency, while the combined CQ and CG properties imply liveness. Following the tradition set by Nakamoto [1], we adopt a continuous-time model and represent PoW mining as a homogeneous Poisson point process. This model was also used in several subsequent influential works [48], [49], [50], [22]. Let λ be the total mining rate of the network, specifically denoting the number of blocks generated within one second. Honest block arrivals and adversarial block arrivals as two independent Poisson processes with rates $\rho\lambda$ and $(1-\rho)\lambda$, respectively. We use λ_i to denote the mining rate of exclusive blocks within shard i. As inclusive blocks extend on the longest chains in all shards, there exists only one mining rate for inclusive blocks spanning all shards, denoted as λ_s. We denote ρ_i as the honest participation ratio within shard i, and Δ_i as the maximum network delay observed within shard i. Below we present the main security result of Manifoldchain.

Theorem 1. *If there is at least one honest miner in each shard, CP, CG, and CQ hold for Manifoldchain regardless of the adversary's attack strategy within each shard i, as long as*

$$p_i = \frac{\lambda_i \rho_i + m\lambda_s \rho}{\lambda_i + m\lambda_s} e^{-(\lambda_i+\lambda_s)\Delta_i} > \frac{1}{2}, \tag{3}$$

The CP property is hold except with a negligible error probability $\varepsilon(\kappa)$ bounded by

$$\varepsilon(\kappa) \leq (2 + 2\sqrt{\frac{p_i}{1-p_i}})(4p_i(1-p_i))^{\kappa}. \tag{4}$$

For any $\delta_i > 0$, $\delta_i' > 0$, the chain growth rate $\mathtt{g}_i$ and chain quality rate $\mathtt{q}_i$ satisfy

$$\begin{aligned} \mathtt{g}_i \geq & (1-\delta_i)\frac{p_i(\lambda_s+\lambda_i)}{1+\Delta_i(\lambda_s+\lambda_i)}, \\ \mathtt{q_i} \geq & 1-(1+\delta_i')\frac{1+\Delta_i p_i \rho_i(\lambda_s+\lambda_i)}{p_i}, \end{aligned} \tag{5}$$

with respective error probabilities of $\varepsilon(\delta_i)$ and $\varepsilon(\delta_i')$.

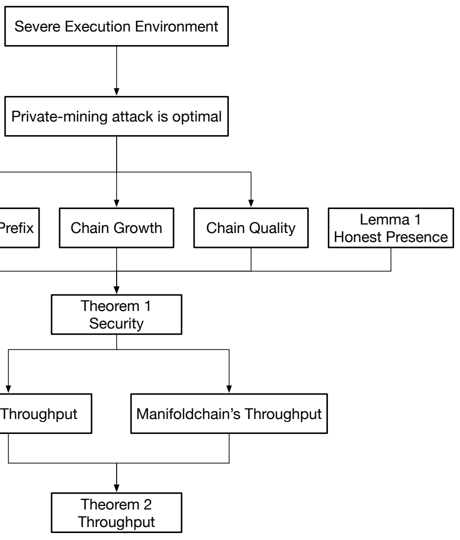

Fig. 5. The proof roadmap of Theorem 1 and 2.

We present the roadmap of our proof in Fig. 5 and defer the detailed proof to Appendix H of the complete version [47]. Specifically, we shift from a standard to a *severe execution environment*, where any delivered full block experiences the maximal network delay, and during the block delivery any mining operation is considered adversarial. Achieving security in severe environment is more challenging than in standard environment. We then establish that the private-mining attack is the optimal strategy in a severe execution environment. Additionally, we calculate the upper bound on mining rates necessary to maintain CP property against the private-mining attack, serving as a constraint against all potential attacks in all environments. Towards CG and CQ properties, we obtain the upper bounds of chain growth speed and honest block fraction based on these bounded mining rates. By leveraging Lemma 1 in conjunction with the CP, CG, and CQ properties, we substantiate the proof of Theorem 1.

This theorem suggests that shards experiencing lower delays can achieve faster mining rates compared to those with higher delays, while still maintaining the same error probability. Assuming all shards share the same κ, according to Eq. 4, there exists a one-to-one mapping between $\varepsilon(\kappa)$ and p_i. An increase in Δ_i also results in a decrease in p_i and $\varepsilon(\kappa)$. Referring to Eq. 3, the maximal values of λ_s and λ_i are constrained by $\varepsilon(\kappa)$. Therefore, shards with lower Δ_i can be configured with higher $\lambda_s+\lambda_i$. With respect to the liveness, the longest chain in shard i achieves an average chain growth rate of $\frac{p_i(\lambda_s+\lambda_i)}{1+\Delta_i(\lambda_s+\lambda_i)}$ and an average adversarial block ratio of $\frac{1+\Delta_i p_i \rho_i(\lambda_s+\lambda_i)}{p_i}$. Consequently, it can be deduced from this theorem that the longest chains in faster shards possesses higher chain growth rates and chain quality ratios over slower shards.

C. Optimal Throughput

Throughput-security trade-off on bandwidth estimation. A larger S_Y helps to better distinguish between fast and slow nodes, thereby enhancing throughput. However, increasing S_Y relies on a more accurate bandwidth estimation. A large estimation error derived from the gap between realistic bandwidth distribution and the estimated one can lead to unevenly distributed honest miners across Y-regions, with some Y-regions having a lower number of honest miners than the average value. This uneven distribution may lead to a larger error probability in Lemma 1.

Optimal mining rate configuration. A key question arises on how to optimally configure λ_s and λ_i to maximize throughput while maintaining security established by Theorem 1. Assuming $\varepsilon(\kappa)$ is regulated to less than ε (We do not consider $\varepsilon(\delta_i)$ and $\varepsilon(\delta_i')$ here because they depend on the Chernoff bound parameters δ_i and δ_i', which are unrelated to the protocol) and κ is fixed to κ'. The network delay within each shard is computed based on the lowest bandwidth within that shard. Specifically, $\Delta_i = \frac{\mathtt{B}}{\underline{C_i}} + \Delta_p$, where $\mathtt{B}$ denotes the block size, $\underline{C_i}$ represents the lowest bandwidth in shard i, and Δ_p is a constant denoting the propagation delay. As the exact honest ratio in a specific shard is generally unknown, we consider the worst case where $\rho_i = 0$, $\forall i$. Consequently, we construct the following optimization problem.

$$\begin{aligned} \max_{\{\lambda_i\}_m,\lambda_s} \quad & \sum_i^m (\lambda_i + \lambda_s) \\ s.t. \quad & p_i = \frac{m\lambda_s\rho}{\lambda_i + m\lambda_s} e^{-(\lambda_i+\lambda_s)\Delta_i} > \frac{1}{2}, \\ & (2 + 2\sqrt{\tfrac{p_i}{1-p_i}})(4p_i(1-p_i))^{\kappa'} \le \varepsilon. \end{aligned} \tag{6}$$

The key challenge to solve this optimization problem lies in how to strike a balance between λ_i's and λ_s, given that both contribute to the overall throughput. We present the algorithm to find the optimal solution for this problem and the key insight here, deferring the detailed proof to Appendix I of the complete paper [47].

- Miners in the slowest shard only mine inclusive blocks (i.e., $\lambda_j = 0$). According to Theorem 1, there exist trade-offs between λ_i/λ_s and the error probability $\varepsilon(\kappa)$ within each shard. Crucially, the trade-off between λ_s and $\varepsilon(\kappa)$ is more favorable than that between λ_i and $\varepsilon(\kappa)$. In other words, for the same increment in λ_s or λ_i, the latter results in a more pronounced increase in $\varepsilon(\kappa)$. Therefore, with a bounded ε, λ_s can be set higher than λ_i. In shard j, $\Delta_j = \Delta$. Given all other known parameters, the maximum value of λ_s, denoted as λ_s', is easily obtained through binary search.
- In other shard i where $\Delta_i < \Delta$, the first constraint of (6) is relaxed, allowing us to set a non-zero λ_i' to achieve an improved throughput.

It is noteworthy that the slowest shard attains the same throughput as Bitcoin. Denote the throughput of Bitcoin as T, we present the total throughput of Manifoldchain as follows:

Theorem 2 (Informal)**.** *Assuming $\forall i, m \gg \frac{\lambda_i}{\lambda_s}$, in a scenario where Bitcoin achieves a throughput of T, Manifoldchain attains a throughput of $\frac{T\Delta}{\rho}\sum_i^m \frac{\rho_i}{\Delta_i}$, while maintaining the same level of security as Bitcoin.*

We present the proof roadmap in Figure 5 and defer the detailed proof to Appendix F-C of the complete paper [47]. Specifically, we calculate the maximum throughput of Bitcoin and Manifoldchain while maintaining identical error probabilities as stated in Theorem 1. By benchmarking Manifoldchain against Bitcoin, we substantiate the proof of Theorem 2.

Given $m \gg \frac{\lambda_i}{\lambda_s}$, in a general scenario where $\forall\, \rho_i = \rho$, Manifoldchain can achieve a throughput of nearly $\frac{\Delta}{\Delta_i}T$ within shard i. In an extreme scenario where the adversary concentrates its hashing power within some shards, and in other shards $\rho_i = 1$, Manifoldchain can achieve a throughput of $\frac{\Delta}{\rho\Delta_i}T$.

VI. Experiments

We implemented a prototype instantiation of a Manifoldchain client in about 15,000 lines of Rust code [23]. In our implementation, a Manifoldchain client employs two parallel threads—one for mining (called *Miner*) and another for delivering incoming blocks from the network (called *Networker*). Both threads have access to two core data structures: *Mempool* and *Multichain*. Below, we outline the specific functionalities of each module:

- **Mempool** receives verified transactions and testimonies from the network module and pairs testimonies with their associated transactions. Upon requested by the Miner module, returns matched transactions and testimonies.
- **Multichain** provides the interface for reading and writing blockchains across all shards, with each individual blockchain maintaining a ledger for a single shard.
- **Miner** retrieves transactions and testimonies from the Mempool, conducts double-spending check, and solves PoW to generate a valid block. Once a valid block is created, it is inserted into Multichain, and also immediately relayed to the Networker module for broadcasting.
- **Networker** manages inter-node communications, employing the Gossip protocol to broadcast and deliver messages. It handles three types of messages: transactions, testimonies, and blocks. Upon receiving messages, it validates their validity before further processing.

With this prototype, we conducted experimental evaluations on the following two different testbeds:

- *Real-world testbed*: We deployed Manifoldchain on Amazon Elastic Compute Cloud (EC2) to evaluate its performance in a genuine geo-distributed environment. Each miner operates within an EC2 `t3.medium` instance outfitted with 2 CPU cores, 4GB of RAM, and a 20GB NVMe SSD. Importantly, these EC2 instances are positioned in 10 major cities geographically distributed worldwide, spanning Sydney, London, Virginia, California, Canada, Ireland, Mumbai, Sao Paulo, Stockholm, and Tokyo. We deploy Manifoldchain in this real-world testbed to evaluate its actual throughput and latency.
- *Simulated testbed*: Due to the complexity of the real-world testbed environment, we additionally deploy Manifoldchain in a simulated testbed to obtain an accurate and reproducible evaluation of its throughput under various

bandwidth settings, while maintaining other configurations unchanged. Specifically, we operate a Manifoldchain system on a single machine with 4 CPU cores, 16GB of RAM, and a 100GB NVMe SSD. We simulate the miners with different processes, adjusting the network bandwidths and delays using tc commands. With this fully controllable tested, we could easily and accurately evaluate Manifoldchain's performance under various network scenarios.

We perform experiments on these testbeds to answer the following questions.

- What is the throughput and the latency of Manifoldchain in a realistic deployment?
- Is Manifoldchain able to achieve better throughput compared with the SOTA full sharding protocol?
- How does the system scale to more miners and higher bandwidths?

To our best knowledge, Monoxide is the only full sharding protocol offering a comparable security level to ours, i.e., 1/2 fault tolerance. Other permissionless sharding protocols, like Elastico, Omniledger, and RapidChain, have fault tolerance less than 1/3 or even 1/4, rendering direct experimental comparisons unfair. We compare with Monoxide to highlight our primary motivation: the throughput can be boost by clustering miners with similar bandwidths. As Monoxide is not open source, we implement it in Rust, and choose the UTXO model instead of its original account/balance model as the transaction model. In this case, Manifoldchain and Monoxide utilizes the same verification scheme to verify both the domestic-txs and cross-txs. Moreover, Manifoldchain and Monoxide have different shard formation mechanisms. Manifoldchain employs the innovative BCSF mechanism proposed in this paper, while Monoxide employs the standard USF mechanism. We evaluate performance of Monoxide on the same testbed as Manifoldchain. In all experiments, all miners share a common block size of approximately 547.14KB (with each block containing 2048 transactions) and maintain a confirmation depth of $\kappa = 6$. The mining rates across shards are configured by setting respective mining difficulties.

A. *Results on Amazon EC2*

Throughput. Our first experiment measures the throughput of Manifoldchain on the realistic testbed. We run Manifoldchain clients on 50 EC2 instances, each hosting a single miner. These instances have been configured with various bandwidth capacities. The available bandwidths are $5, 10, 20, 40, 60$ Mbps, with each bandwidth being adopted by 10 miners. We distribute these 50 miners into 5 shards, with 10 miners in each shard. As Manifoldchain and Monoxide adopts different shard formation mechanisms, they exhibit different bandwidth distributions across shards. Specifically,

- Monoxide adopts USF mechanism:
  - Shard 0: $\{5, 5, 10, 10, 20, 20, 40, 40, 60, 60\}$.
  - Shard 1: $\{5, 5, 10, 10, 20, 20, 40, 40, 60, 60\}$.
  - Shard 2: $\{5, 5, 10, 10, 20, 20, 40, 40, 60, 60\}$.
  - Shard 3: $\{5, 5, 10, 10, 20, 20, 40, 40, 60, 60\}$.
  - Shard 4: $\{5, 5, 10, 10, 20, 20, 40, 40, 60, 60\}$.
- Manifoldchain adopts BCSF mechanism:
  - Shard 0: $\{5, 5, 5, 5, 5, 5, 5, 5, 5, 5\}$.
  - Shard 1: $\{10, 10, 10, 10, 10, 10, 10, 10, 10, 10\}$.
  - Shard 2: $\{20, 20, 20, 20, 20, 20, 20, 20, 20, 20\}$.
  - Shard 3: $\{40, 40, 40, 40, 40, 40, 40, 40, 40, 40\}$.
  - Shard 4: $\{60, 60, 60, 60, 60, 60, 60, 60, 60, 60\}$.

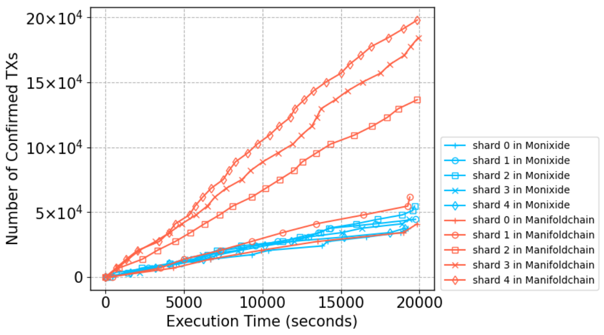

Fig. 6. Throughputs of Manifoldchain and Monoxide in distinct shards. Comprising stragglers, all shards in Monoxide and the slowest shard in Manifoldchain achieves nearly the same throughput. Other shards in Manifoldchain, free from stragglers, achieve faster throughput.

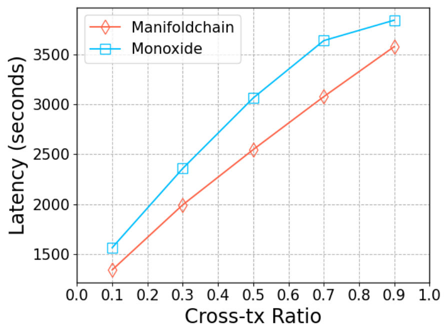

Fig. 7. Average transaction latency for different cross-tx ratios. Manifoldchain consistently exhibits lower latency than Monoxide. As the cross-tx ratio increases, latency rises in both Manifoldchain and Monoxide.

We set the mining rates λ_s and λ_i within each shard by solving Optimization 6, with the same target error probability for Manifoldchain and Monoxide. The comparable security levels of these two sharding protocols are evidenced by their similar forking rates, both measured at less than 3% in the experiment. As shown in Figure 6, across all shards, Manifoldchain achieves an average throughput of 30.41 tx/sec, while Monoxide only achieves an average throughput of 10.62 tx/sec. The gain in throughput of Manifoldchain is attributed to improved performance in the fast shards where miners are not compromised by stragglers: the throughput in the fastest shard is 5.11 times that of the slowest shard.

Latency. Under the same parameter setting, we evaluate the confirmation latency of Manifoldchain and compare it with Monoxide. We manually generate domestic-tx as well as cross-txs and broadcast them within the corresponding shards. A domestic-tx is randomly assigned to one shard, while a cross-tx is randomly assigned to four shards. Setting $\kappa = 6$ across all shards, we evaluate the average latency of Manifoldchain

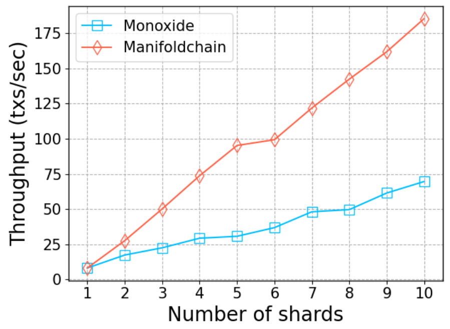

Fig. 8. Throughput scaling with the growing number of shards. Manifoldchain demonstrates a more rapid increase in throughput compared with Monoxide.

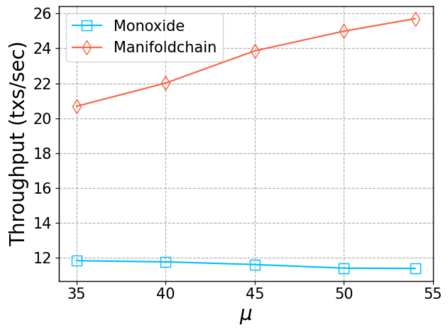

Fig. 9. Throughput scaling with the average bandwidth of normal miners (μ). Manifoldchain achieves higher throughput as μ increases, whereas Monoxide maintains nearly constant throughput across varying μ.

and Monoxide over all shards under different cross-tx ratios ranging from 0.1 to 0.9. Figure 7 shows that Manifoldchain achieves an overall lower latency than Monoxide. It is because that Manifoldchain generally has faster mining rates, and reaches the target confirmation depth with less time. As cross-txs rely on confirmations of all input-txs and output-txs which may locate in slow shards, their confirmation time is longer than that of domestic-txs. Consequently, as the cross-tx ratio increases, the average latency also rises.

B. Results on Simulated Testbed

The following experiments are conducted on our simulated testbed, to demonstrate the horizontal and vertical scalability of Manifoldchain, and compare them with Monoxide.

Horizontal scalability. We set the size of each shard to 5 and progressively increase the number of shards m. To simulate a scenario with varying bandwidths, we have 5 miners configured with a bandwidth configuration of $\{5, 10, 20, 40, 60\}$ Mbps. For each increment in m, we add 5 miners with this bandwidth configuration. Subsequently, we measure the maximum throughput achieved by Manifoldchain and Monoxide under varying values of m.

Fig. 8 demonstrates the horizontal scalability of Manifoldchain and Monoxide. Manifoldchain achieves an average increase in throughput of 19.68 tx/sec for each increment in m, while Monoxide can only achieve an average increase in throughput of 6.83 tx/sec under the same increment. The reason why Manifoldchain achieves better horizontal scalability is that it fully utilizes the fast miners' bandwidths through BCSF mechanism for each shard increment. In Monoxide, the newly joined miners contain both stragglers and fast miners. The stragglers are uniformly distributed across shards. Therefore, regardless of the shard to which the fast miners are allocated, they are compromised by the presence of stragglers. In sharp contrast, in Manifoldchain, the stragglers among the new miners are assigned to the slow shards, while the fast miners are allocated to the fast shards, admitting higher mining rates. Therefore, the throughput increase results from each shard increment depends on the overall bandwidths configured by all the newly joined miners, which is larger than that in Monoxide.

Vertical scalability. The vertical scalability reflects in how the throughput scales with the increase in the bandwidths of normal miners while considering the presence of stragglers. We simulate this scenario by sampling bandwidths of stragglers and normal miners from two distinct normal distributions. Specifically, there are a total of 25 miners, with 20 being normal miners and the remaining 5 miners being stragglers. The stragglers follow a normal distribution $\mathcal{N}_1(10, 2)$, and the normal miners follow a normal distribution $\mathcal{N}_2(\mu, 7)$, where μ varies from 35 to 54. These 25 miners are distributed across 5 shards, with 5 miners in each shard. In this scenario, we evaluate the maximum throughput achievable by Manifoldchain and Monoxide under varying values of μ.

As shown in Figure 9, Manifoldchain achieves an average increase in throughput of 1.25 tx/sec for each increment in μ, while Monoxide merely achieves a constant throughput under different μ. The absence of vertical scalability in Monoxide can be attributed to the presence of stragglers within each shard. Since the stragglers are uniformly distributed across shards, they compromise nearly all the normal miners. Despite the increase in the bandwidths of normal miners, the mining rates cannot be correspondingly increased. In contrast, normal miners in Manifoldchain are not compromised by the presence of stragglers. If the bandwidths of normal miners increase, they can be configured with a higher mining rate to achieve higher throughput while keeping the error probability unchanged.

VII. Conclusion and Discussions

We propose Manifoldchain, a permissionless full sharding protocol which boost vertical scalability within each shard via clustering miners with similar bandwidths. This uneven shard partition may introduce a new vulnerability: the adversary could concentrate hashing power in one shard to establish an adversarial majority. To counter this attack, we offer sharing mining to diffuse the honest hashing power to the entire network, thereby ensuring each shard attains the same level of security as an unsharded blockchain. This design positions Manifoldchain as the most robust sharding protocol, achieving optimal fault tolerance within each shard: security is upheld as long as there is at least one miner in each shard. Besides, we provide a tight security analysis that guides us to set the optimal mining rates across shards to maximize throughput. Furthermore, we implement Manifoldchain, and evaluate its performance on both realistic and simulated testbeds. The experimental results not only validate our theoretical assertions but also illustrate that Manifoldchain delivers a notable improvement in throughput over SOTA full sharding protocols.

Adapting to non-PoW consensus. Manifoldchain's core design principle is versatile, enabling straightforward extension to non-PoW sharding protocols, notably Proof-of-Stake (PoS) and permissioned settings. In PoS systems, uneven stake distribution poses a risk of adversarial dominance within shards. Sharing mining can mitigate this by allowing miners to use a single coin to mine across shards, thus dispersing their power to secure potentially vulnerable shards. On the other hand, incorporating sharing mining into permissioned sharding protocols resemble the design of *shared security* in the Cosmos ecosystem [51], where different Cosmos chains share a common set of validators. We defer a detailed comparison and analysis to future work.

Achieving optimal vertical scalability. Manifoldchain improves horizontal and vertical scalability by clustering nodes with similar bandwidth within the same shard. This idea could be synergistically integrated with protocols optimizing vertical scalability, such as Prism [25] and recent DAG-based BFT protocols [52], [53], [54], [55]. These protocols achieve high throughput, approaching the network capacity, by decoupling block proposal and transaction validation. Incorporating these protocols as the intra-shard consensus, Manifoldchain stands to optimize both vertical and horizontal scalability. Specifically, integrating Prism into Manifoldchain is straightforward: the proposer blocks in Prism can be substituted with exclusive/inclusive blocks, following the same mining and verification processes as Manifoldchain. A detailed analysis of this integration is earmarked for future exploration.

Acknowledgment

This work is supported in part by the National Nature Science Foundation of China (NSFC) Grant 62106057, the Fundamental Research Funds for the Central Universities (Grant No. 2242024k30059), a gift from Stellar Development Foundation, and the Guangzhou-HKUST(GZ) Joint Funding Program (No. 2024A03J0630). We are also grateful to Lei Yang (MIT) for his valuable suggestions on our evaluation. We also thank our shepherd and all anonymous reviewers for their helpful feedback.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[2] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[3] Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. How does nakamoto set his clock? full analysis of nakamoto consensus in bounded-delay networks. *Cryptology ePrint Archive*, 2020.

[4] Charts. Bitcoin network hash rate. https://ycharts.com/indicators/bitcoin-network-hash-rate.

[5] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[6] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[7] Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pages 95–112, 2019.

[8] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[9] Ranvir Rana, Sreeram Kannan, David Tse, and Pramod Viswanath. Free2shard: Adversary-resistant distributed resource allocation for blockchains. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(1):1–38, 2022.

[10] Adem Efe Gencer, Robbert van Renesse, and Emin Gün Sirer. Service-oriented sharding with aspen. *arXiv preprint arXiv:1611.06816*, 2016.

[11] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. Tor: The second-generation onion router. In *USENIX security symposium*, volume 4, pages 303–320, 2004.

[12] Karsten Loesing. Measuring the tor network from public directory information. *Proc. HotPETS, Seattle, WA*, 2009.

[13] Mashael Alsabah and Ian Goldberg. Performance and security improvements for tor: A survey. *ACM Comput. Surv.*, 49(2), sep 2016.

[14] Andriy Panchenko, Fabian Lanze, and Thomas Engel. Improving performance and anonymity in the tor network. In *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*, pages 1–10, 2012.

[15] Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 683–696, New York, NY, USA, 2022. Association for Computing Machinery.

[16] Yibin Xu, Jingyi Zheng, Boris Düdder, Tijs Slaats, and Yongluan Zhou. A two-layer blockchain sharding protocol leveraging safety and liveness for enhanced performance. *IACR Cryptol. ePrint Arch.*, page 304, 2024.

[17] Jianting Zhang, Wuhui Chen, Sifu Luo, Tiantian Gong, Zicong Hong, and Aniket Kate. Front-running attack in sharded blockchains and fair cross-shard consensus, 2023.

[18] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 114–134. Springer, 2020.

[19] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. *arXiv preprint arXiv:1809.09044*, 2018.

[20] James N Gray. Notes on data base operating systems. *Operating systems: An advanced course*, pages 393–481, 2005.

[21] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer, 2015.

[22] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.

[23] Manifoldchain. Rust implementation of the manifoldchain sharding protocol. https://github.com/Hide-on-bush2/Manifoldchain.

[24] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In Katerina J. Argyraki and Rebecca Isaacs, editors, *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, pages 45–59. USENIX Association, 2016.

[25] Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 585–602. ACM, 2019.

[26] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[27] Jelle Hellings and Mohammad Sadoghi. Byshard: sharding in a byzantine environment. *VLDB J.*, 32(6):1343–1367, 2023.

[28] Dongning Guo and Ling Ren. Bitcoin's latency-security analysis made simple. *CoRR*, abs/2203.06357, 2022.

[29] Peter Gazi, Ling Ren, and Alexander Russell. Practical settlement bounds for proof-of-work blockchains. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1217–1230. ACM, 2022.

[30] Jing Li, Dongning Guo, and Ling Ren. Close latency-security trade-off for the nakamoto consensus. In Foteini Baldimtsi and Tim Roughgarden, editors, *AFT '21: 3rd ACM Conference on Advances in Financial Technologies, Arlington, Virginia, USA, September 26 - 28, 2021*, pages 100–113. ACM, 2021.

[31] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017.

[32] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks. In Amar Phanishayee and Vyas Sekar, editors, *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*, pages 493–512. USENIX Association, 2022.

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

[34] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 507–527. Springer, 2015.

[35] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, pages 1–10. IEEE, 2013.

[36] Yuan Wang, Hideaki Ishii, François Bonnet, and Xavier Défago. Resilient real-valued consensus in spite of mobile malicious agents on directed graphs. *IEEE Transactions on Parallel and Distributed Systems*, 33(3):586–603, 2022.

[37] Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 412–456. Springer, 2020.

[38] Marek Klonowski, Dariusz R. Kowalski, and Jaroslaw Mirek. Ordered and delayed adversaries and how to work against them on a shared channel. *Distributed Comput.*, 32(5):379–403, 2019.

[39] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. *Distributed Comput.*, 36(1):3–28, 2023.

[40] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.

[41] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53, oct 2016.

[42] Manish Jain and Constantinos Dovrolis. End-to-end estimation of the available bandwidth variation range. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, page 265–276, New York, NY, USA, 2005. Association for Computing Machinery.

[43] Peiyao Sheng, Nikita Yadav, Vishal Sevani, Arun Babu, S. V. R. Anand, Himanshu Tyagi, and Pramod Viswanath. Proof of backhaul: Trustfree measurement of broadband bandwidth. *CoRR*, abs/2210.11546, 2022.

[44] Idit Keidar and Danny Dolev. Increasing the resilience of atomic commit at no additional cost. In Mihalis Yannakakis and Serge Abiteboul, editors, *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 245–254. ACM Press, 1995.

[45] Sung-Hoon Park, Jea-Yep Lee, and Su-Chang Yu. Non-blocking atomic commitment algorithm in asynchronous distributed systems with unreliable failure detectors. In Shahram Latifi, editor, *Tenth International Conference on Information Technology: New Generations, ITNG 2013, 15-17 April, 2013, Las Vegas, Nevada, USA*, pages 33–38. IEEE Computer Society, 2013.

[46] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[47] Chunjiang Che, Songze Li, and Xuechao Wang. Manifoldchain: Maximizing blockchain throughput via bandwidth-clustered sharding. *arXiv preprint arXiv:2407.16295*, 2024.

[48] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.

[49] Ling Ren. Analysis of nakamoto consensus. *Cryptology ePrint Archive*, 2019.

[50] Jing Li and Dongning Guo. Continuous-time analysis of the bitcoin and prism backbone protocols. *arXiv preprint arXiv:2001.05644*, 2020.

[51] Christina Cosmos. Interchain security begins a new era for cosmos. https://blog.cosmos.network/interchain-security-begins-a-new-era-for-cosmos-a2dc3c0be63.

[52] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.

[53] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 165–175, 2021.

[54] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2718, 2022.

[55] Alexander Spiegelman, Balaji Aurn, Rati Gelashvili, and Zekun Li. Shoal: Improving dag-bft latency and robustness. *arXiv preprint arXiv:2306.03058*, 2023.

Appendix A
Adaptive Adversary

Adaptive adversaries are further categorized into three types:

1) *Mildly* adaptive [6]: When the adversary requests to corrupt a miner, the corruption occurs after a certain delay.
2) *Weakly* adaptive [37], [38]: When the adversary requests to corrupt a miner, the corruption occurs after the miner sends all outgoing messages. Additionally, any messages already sent by the miner cannot be erased by the adversary.
3) *Strongly* adaptive [37], [39]: When the adversary requests to corrupt a miner, the corruption occurs immediately. Furthermore, any messages sent by the miner before the corruption that have not yet been arrived can be erased by the adversary.

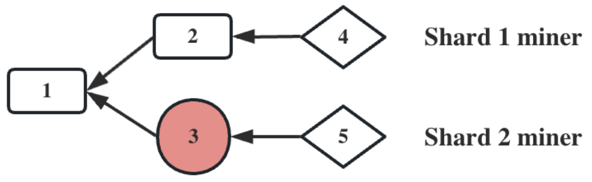

Fig. 10. Hashing power splitting attack. The adversary mines an exclusive block 3 with an associated invalid transaction block in shard 1. The exclusive block is deemed invalid in shard 1 but regarded as valid in other shards.

Appendix B
Hashing Power Splitting Attack

We define the hashing power splitting attack (HPSA) as follows:

Definition 2. *(Hashing Power Splitting Attack.) Adversary mines an exclusive block with an invalid transaction block and send the exclusive block to other shards, splitting the hashing power of honest miners.*

As shown in Fig 10, a corrupted miner in shard 1 mines an exclusive block 3 containing an invalid transaction block. In the initial design of sharing mining, only block headers are broadcast across shards, thus miners can only verify transactions within their own shard. In shard 1, block 3 is invalid, but the adversary sends it to shard 2, where it is deemed valid because shard 2 miners do not verify the transaction. If an inclusive block 5 is mined on top of block 3, shard 1 miners reject this chain and continue mining on blocks 2 and 4. This attack splits the hashing power of honest miners across different chains.

Under the worst-case scenario, HPSA could lead to a consensus failure in a shard with an adversarial majority. This attack entails an adversary with overwhelming hashing power mining a longer chain that consists of all invalid transaction blocks and making it public to the global view. In this scenario, honest miners would consider the adversarial chain as valid because they are unable to verify the availability and validity of transactions since only the block headers are accessible. Therefore, other miners across shards cannot access and collaborate on the susceptible honest chain in the source shard, resulting in their inability to contribute their hashing power to the honest miners in that shard. Consequently, without honest majority and validity/availability verification, the adversary could arbitrarily cause double-spending failures.

Appendix C
Transaction Validity & Block Availability Verification

Transaction validity verification. In a target shard where the adversary aims to split hashing power, miners within the shard are "in-shard miners", and those outside are "out-shard miners". Out-shard miners rely on at least one honest in-shard miner to verify transaction validity by sending a fraud proof, denoted as $\texttt{proof}_{\texttt{o}}\texttt{f}_{\texttt{i}}\texttt{nvalidity}$, formatted as follows:

$$\begin{aligned}&\texttt{proof\_of\_invalidity} = \\ &\quad\{(\texttt{tx}_0, \texttt{block\_hash}_0, \texttt{tx\_merkle\_proof}_0), \\ &\quad(\texttt{tx}_1, \texttt{block\_hash}_1, \texttt{tx\_merkle\_proof}_1), \\ &\quad\texttt{fault\_type}\}\end{aligned} \tag{7}$$

For double spending faults, the fraud proof shows the invalidity of a transaction block by proving the existence of two conflicting transactions using the same UTXOs. $\texttt{tx}_0$ is a transaction in the invalid block with $\texttt{block\_hash}_0$ as the hash of its consensus block, verifiable with $\texttt{tx\_merkle\_proof}_0$. $\texttt{tx}_0$ is another transaction conflicting with $\texttt{tx}_1$, linked to $\texttt{block\_hash}_1$, and verified with $\texttt{tx\_merkle\_proof}_1$. For incorrect formatting faults, such as invalid signatures, only the inclusion proof of $\texttt{tx}_0$ is required, and the second component of the fraud proof is left empty. Out-shard miners verify the fraud proof; if valid, they mark the block as verified; otherwise, they mark it as invalid.

Data availability verification. We utilize Coded Merkle Tree (CMT) proposed by Yu et al. [18] to facilitate the data availability verification across shards. Specifically, different miners' behaviours are summarized as follows:

1) Block producer (in-shard miner): (a) Generates a consensus block (which may be either an exclusive or inclusive block) containing CMT and broadcasts this block to all miners, while broadcasting the associated transaction block exclusively to in-shard miners. (b) Responds to sample requests received from out-shard miners.
2) Out-shard miner: (a) Upon receiving a new consensus block from other shards, initiates separate, anonymous, and intermittent sampling requests with replacement directed at in-shard miners who claim the block is available. (b) Broadcasts received samples to all connected in-shard miners. (c) Assumes block availability if all requested samples are received. (d) Rejects the consensus block if any requested samples are not received within the bounded network delay Δ. (e) Rejects the block if any error proofs, such as `incorrect_coding_proof` or `bad_code_proof`, are received.
3) Other in-shard miner: (a) Upon receiving valid samples, attempts to recover the data block through both downloading the associated transaction block from other in-shard miners and collecting samples forwarded by out-shard miners. (b) Rebroadcasts received valid samples. (c) Sends corresponding proof and rejects the block if adversarial behavior, such as incorrect coding or bad code, is detected. (d) Declares the availability of the block to all other miners and responds to sample requests from out-shard miners upon successful receipt or full decoding and verification of a data block.

Appendix D
Cross-shard Transaction Atomicity

In this section,we prove the cross-shard transaction atomicity of Manifoldchain. To facilitate the proof, we present the following lemma:

Lemma 2. *In Manifoldchain, all input-txs, output-txs, refund-txs created by honest users will eventually be confirmed.*

Proof. Honest users always faithfully follow the protocol to broadcast input-txs, output-txs, and refund-txs in all involved shards. Due to the liveness of Manifoldchain, these transactions will eventually be confirmed. □

Formally, we have the following theorem:

Theorem 3 (Cross-shard transaction atomicity). *In Manifoldchain, for a cross-tx created by honest users, (1) if all input-txs are confirmed as accepted, then all output-txs will eventually be confirmed as accepted; (2) if any input-tx is confirmed as rejected, then all output-txs will eventually be confirmed as rejected, and all refund-txs will eventually be confirmed as accepted.*

Proof. We will prove statement (1) by contradiction. Suppose that all input-txs are confirmed as accepted and Lemma 2 holds. For (1) to be false, there must exist an output-tx that is confirmed as rejected. However, according to the verification mechanism described in Section IV-D, a confirmed rejected input-tx must exist in at least one input shard for an output-tx to be confirmed as rejected. This directly contradicts the initial condition that all input-txs are confirmed as accepted, as an input-tx can only be confirmed as either accepted or rejected, not both.

We will also prove statement (2) by contradiction. Suppose that there exists an input-tx that is confirmed as rejected and Lemma 2 holds. For (2) to be false, at least one of the following events must occur: (i) at least one output-tx is confirmed as accepted, or (ii) at least one refund-tx is confirmed as rejected. Event (i) necessitates that all input-txs are confirmed as accepted, which directly contradicts the condition that one of the input-txs is confirmed as rejected, as an input-tx can only be confirmed as either accepted or rejected. Therefore, event (ii) must also be false, as it requires at least one output-tx to be confirmed as accepted, which has already been proven false. □

Appendix E
Security Properties

In this section, we present the formal definitions of security properties. We denote by $\mathcal{C}_i^t$ the longest chain in miner i's view at time t.

A. Common Prefix

CP property stipulates that honest miners should agree on the current chain, with the exception of a small number, κ, of unconfirmed blocks at the end of the chain. Prior to the definition of CP, we introduce the following bad event E_v:

Definition 3. ***(Consistency violation of a target transaction)****. The consistency of a target transaction is violated when a block containing the target transaction is confirmed, while simultaneously, another distinct block (possibly containing or not containing the target transaction) is confirmed at the same height within the blockchain.*

We define CP property $\mathcal{CP}(\kappa, \varepsilon(\kappa))$ as follows:

Definition 4. *A protocol $\prod$ holds common prefix property $\mathcal{CP}(\kappa, \varepsilon(\kappa))$ in environment $\mathcal{E}nv(\Delta, \rho)$ if consistency violation events happen with an negligible probability $\varepsilon(\kappa)$ when employing κ-confirmation rule in case that any message is delayed for Δ seconds and the ratio of the adversarial hashing power is ρ.*

B. Chain Growth

CG property stipulates that in a honest miner's view, the length of the longest chain should grow at an average rate g over a continuous time span of $\mathcal{T}$ seconds, except with an exponentially small probability $\varepsilon(\mathcal{T})$. Prior to the definition of CG, we introduce the following bad event E_g:

Definition 5. *We say the anticipated growth event $E_g(t, \Delta, \mathcal{T})$ occurs, denoted by $E_g(t, \Delta, \mathcal{T}) = 1$, if the following two events hold true:*

- ***(Consistent Length.)*** *Given the current time $\mathtt{t}$, For any time $r \leq \mathtt{t} - \Delta$, $r + \Delta \leq r' \leq \mathtt{t}$, for every two miners i, j such that i is hones at r and j is honest at r', $len(\mathcal{C}_j^{r'}) \geq len(\mathcal{C}_i^r)$ holds.*
- ***(Chain Growth.)*** *For any time $r \leq \mathtt{t} - t$, it holds that $\min_{i,j}(len(\mathcal{C}_j^{r+t}) - len(\mathcal{C}_i^r)) \geq \mathcal{T}$.*

The CG property is defined as follows:

Definition 6. *A blockchain protocol $\prod$ holds chain growth property $\mathcal{CG}(\mathtt{g}, \varepsilon(\cdot))$ in environment $\mathcal{E}nv(\Delta, \rho)$ if anticipated growth event E_g happens except with a negligible probability $\varepsilon(\mathcal{T})$. Formally, there exists a constant c such that for ever $\mathcal{T} \geq c$ and $t \geq \frac{\mathcal{T}}{\mathtt{g}}$, the following holds:*

$$\mathtt{Pr}[E_g(t, \Delta, \mathcal{T}) = 1] \geq 1 - \varepsilon(\mathcal{T}). \tag{8}$$

C. Chain Quality

In essence, the CQ property necessitates that any longest chain, as adopted by honest miners, must contain a certain proportion of honest blocks. Prior to the definition of CQ, we introduce the following bad event E_q:

Definition 7. *We say the anticipated quality event $E_q(\mathtt{q}, \mathcal{T})$ occurs, denoted by $E_q(\mathtt{q}, \mathcal{T}) = 1$, provided that for each moment t and every honest miner i, within any continuous sequence of $\mathcal{T}$ blocks in $\mathcal{C}_i^t$, the proportion of honest blocks is at a minimum of $\mathtt{q}$.*

The CQ property is defined as follows:

Definition 8. *A blockchain protocol $\prod$ holds chain quality property $\mathcal{CQ}(\mathtt{q}, \varepsilon(\cdot))$ in environment $\mathcal{E}nv(\Delta, \rho)$ if anticipated quality event E_q occurs except with the negligible probability $\varepsilon(\mathcal{T})$. Formally, there exists a constant c such that for every $\mathcal{T} \geq c$, the following holds:*

$$\mathtt{Pr}[E_q(\mathtt{q}, \mathcal{T}) = 1] \geq 1 - \varepsilon(\mathcal{T}). \tag{9}$$