CoSpaN: Permissionless Consensus in Sparse Networks

Thang N. Dinh[1], Jonathan Katz[2], Phuc D. Thai[3], and Hong-Sheng Zhou[1,4]

[1] Virginia Commonwealth University, Richmond, USA
{tndinh,hszhou}@vcu.edu
[2] Google, Washington DC, USA
jkatz2@gmail.com
[3] Sky Mavis, Ho Chi Minh City, Vietnam
phuc.thai@skymavis.com
[4] Logos Blockchain, Institute of Free Technology
hong-sheng@status.im

Abstract. Security analyses of Nakamoto-style blockchains typically assume reliable dissemination, yet real peer-to-peer overlays are sparse and vulnerable to eclipse and Sybil attacks. Existing rigorous dissemination protocols either rely on publicly known resources or do not directly address the proof-of-work (PoW) setting.

In this paper, we present CoSpaN, a protocol for sparse PoW networks that uses merge-mined registrations to obtain publicly verifiable evidence of mining activity and uses VRFs to construct a sparse core-periphery overlay whose expected connectivity is proportional to mining power. Rather than exposing a globally visible set of highly connected nodes, CoSpaN limits broad exposure of the mapping between logical core identities and network addresses, thereby making targeted DoS attacks substantially harder.

We formalize consensus in sparse networks, state an explicit bootstrapping condition for the initial epoch, and analyze CoSpaN against static, mildly adaptive, and slow-observation adaptive adversaries under explicit delay assumptions. These adaptive theorems should be read as model-conditional guarantees for delayed-reaction adversaries, not as claims against the strongest open-network attacker. We also discuss protocol overhead and justify the main parameter choices used in our numerical evaluation. Our numerical results indicate that CoSpaN achieves substantially stronger Sybil robustness than Bitcoin-like sparse overlays at comparable sparsity.

1 Introduction

Bitcoin [18] relies on a peer-to-peer (P2P) overlay to propagate blocks among miners. Standard security analyses assume reliable dissemination, namely that a valid message known to one honest node reaches all honest nodes within bounded time [9,21]. In practice, however, blockchain overlays are sparse: Bitcoin nodes keep only a small number of long-lived connections, e.g., 8 outbound and at most

125 inbound connections [5,19], and many retain the same peers for hours [2]. This sparsity makes reliable dissemination vulnerable to eclipse- and Sybil-style attacks [12,25,16,11,20].

1.1 Reliable dissemination

Eclipse attacks. An adversary can isolate a target by monopolizing its outbound and inbound links. Outbound monopolization can be induced by luring honest nodes toward adversarial peers [12]; inbound monopolization can be achieved via connection-starvation attacks that fill the target's inbound slots with Sybil nodes [6,12]. In sparse networks, a fully adaptive adversary that instantly corrupts nodes and learns the topology can in principle eclipse any node of degree at most f by corrupting all its neighbors [8,24,13].

Protocol	PoW	DoS resilience	Reliable dissemination	Verifiable random connection	Sparse network	Non-mining nodes
[15]	✗	✗	✓	✗	✗	✗
[14]	✗	✗	✓	✗	✗	✓
[3]	✗	✗	✗	✓	✓	✗
CoSpaN	✓	✓	✓	✓	✓	✓

Table 1: Comparison with prior rigorous dissemination designs.

Prior rigorous designs. Recent work studies dissemination against delayed adaptive adversaries [17,15,3,14]. The expander-based designs of [15,14] mitigate outbound monopolization but do not verifiably constrain inbound connections; they also rebuild a new random graph per message, which is not sparse over long periods. Generals' Scuttlebutt [3] gives verifiable stake-weighted connections in the proof-of-stake setting.

PoW-specific challenge. Those designs assume publicly known resources. In PoW, mining power is not directly verifiable, and publicly exposing resource-rich endpoints would also make highly connected nodes easier to target with DoS attacks.

Related work on miner-proportional secondary artifacts. The idea of deriving miner-proportional secondary artifacts from the underlying proof-of-work process has prior precedent, including FruitChains [22], Subchains [23], and other secondary-PoW / merge-mining-style mechanisms [10]. Our contribution is not the existence of such artifacts by itself, but how to use them to build a sparse, verifiable, and selectively exposed dissemination overlay for PoW together with a consensus analysis that explicitly models sparse communication and delayed-reaction adversaries. A 2-for-1 PoW primitive could replace our merge-mined registrations without materially changing the overlay construction or its analysis; we use merge-mined registrations because they keep a simple Nakamoto-style interface and fit naturally with our epoch-validity rules.

1.2 Our results

We present CoSpaN, a sparse-overlay protocol for PoW networks. Nodes use merge-mined registrations as publicly verifiable evidence of mining activity, and each registration acts as a logical core identity in the next epoch. VRFs then determine verifiable random core-core and core-periphery connections, so the expected number of connections obtained by a node is proportional to its mining power.

Restricted exposure of core nodes. Rather than claiming perfect anonymity for highly connected nodes, CoSpaN limits broad exposure of the mapping between logical core identities and physical addresses. Only selected connection counterparts learn the address needed for that specific connection, which makes large-scale targeted DoS attacks substantially harder while matching the actual mechanism more faithfully.

Model and analysis. We formalize consensus in sparse networks, including an explicit one-time bootstrapping condition for the first epoch. We analyze CoSpaN against static, mildly adaptive, and slow-observation adaptive adversaries under explicit delay assumptions. These adaptive theorems are conditional guarantees for delayed-reaction adversaries, not claims against the strongest open-network attacker with immediate traffic visibility.

Comparison and evaluation. Table 1 summarizes the qualitative comparison with prior designs: unlike earlier rigorous protocols, CoSpaN directly addresses PoW, maintains sparse long-lived overlays, supports non-mining nodes, and uses verifiable random connections while limiting public exposure of high-connectivity endpoints. We also discuss computation, communication, and on-chain overhead, and justify the main parameter choices used in our numerical evaluation. Our experiments indicate that CoSpaN offers substantially stronger Sybil robustness than Bitcoin-like sparse overlays at comparable sparsity.

Interpreting the adaptive results. The adaptive models in this paper are motivated by the impossibility of reliable dissemination in sparse networks against an adversary that can both react instantly and learn the relevant topology immediately. In that sense, the delayed-reaction theorems are not meant to claim universal robustness for arbitrary open P2P deployments; rather, they identify explicit regimes in which sparse PoW overlays can still provide rigorous guarantees, and they make clear which reaction-time assumptions are doing the work. These theorems do not relax the underlying delay requirements or suggest that such requirements are mild in every deployment. Instead, they expose explicitly which timing hypotheses the proofs use and where the impossibility boundary reappears once reaction and observation become effectively instantaneous.

Organization. Section 2 introduces new models for consensus in sparse networks. Section 3 presents the design of the CoSpaN protocol. Section 4 gives the security analysis under the static model and the stated weakly adaptive delayed-reaction models. Section 5 reports the numerical studies. Supplemental material for those sections can be found in Appendix.

2 Consensus Models for Sparse Networks

In this section, we introduce our network-aware consensus model, which captures the *sparse* nature of real-world P2P networks. We further define the formal concept of *reliable dissemination* and the task of designing *consensus in sparse networks*, providing the first theoretical framework for analyzing consensus protocols in sparse networks. (Additional information is provided in Appendix A.)

2.1 Sparse network model (SNM)

We introduce a *sparse network model* (SNM) for a blockchain protocol execution. Extending the model by Pass, Seeman, and Shelat [21], referred to as the PSS model, our model includes a set N that consists of n ($n \in \mathbb{N}$) participating nodes, a blockchain protocol (Π, Λ), an environment $\mathcal{Z}$ that captures all aspects external to the protocol itself, and an adversary $\mathcal{A}$. The execution proceeds in *rounds* that model time steps.

There are three major differences between the SNM model and the PSS model. First, we consider a *point-to-point communication* in which nodes can only send messages after establishing connections to other nodes. This contrasts with the PSS model, which considers a communication model in which nodes can send messages to all other nodes. Secondly, we consider a *non-flat* mining model where each node can have a different amount of mining power, in contrast to the flat mining model (i.e., all nodes have the same mining power) in PSS. In particular, our non-flat model allows *nodes with zero-mining power*, e.g., nodes that participate in relaying messages but not the mining process. Such nodes and their corruption were not investigated in the PSS model. We are the first to *go beyond the standard corruption of the mining nodes and investigate the corruption of zero-mining power nodes that are responsible for communication functionality*. Finally, in the SNM model, we start with a *static adversary* that controls the same subset of nodes during protocol execution. In the full version we consider two (weakly) *adaptive adversaries* who may corrupt honest nodes during the protocol execution but in certain restricted manners.

Participating nodes. The set of participating nodes N can be partitioned into the set of *honest nodes* $\mathsf{H} \subset \mathsf{N}$, who strictly follow the blockchain protocol (Π, Λ), and the set of *malicious nodes* $\mathsf{N} \setminus \mathsf{H}$, which are controlled by an *adversary* $\mathcal{A}$. The number of malicious nodes $f = |\mathsf{N} \setminus \mathsf{H}|$ is bounded by $\eta \cdot n$ for a fixed $\eta \in (0, 1)$.

A blockchain protocol. Algorithm Π, which takes a security parameter κ as input, dictates how each node communicates and maintains a blockchain data structure $\mathcal{C}$, a collection of blocks. Algorithm Λ contains a set of rules to extract a ledger $\mathcal{L}$ from $\mathcal{C}$, i.e., $\mathcal{L} = \Lambda(\mathcal{C})$. Here, the ledger $\mathcal{L}$ is a sequence of transactions, i.e., $\mathcal{L} = tx_1 \| tx_2 \| \ldots \| tx_i \| \ldots$. We also overload the notation Λ to define the position of a transaction tx in the ledger. Specifically, for a transaction tx, $\Lambda(\mathcal{C}, tx) = i$ if $tx = tx_i$, the i-th transaction in the ledger, and $\Lambda(\mathcal{C}, tx) = 0$ if tx is not included in the ledger. The validity of a ledger is captured by a predicate

V. The predicate $\mathsf{V}(\mathcal{L})$ returns 1 if and only if the ledger $\mathcal{L}$ is *valid* for some notion of validity (e.g., no double spending). The algorithm Π is parameterized by V (denoted by Π^{V}) to ensure that nodes always maintain valid ledgers.

Non-flat mining model. All nodes interact with a random oracle $H : \{0,1\}^* \to \{0,1\}^\kappa$ (where κ is the security parameter) that takes an arbitrary length string and output a random value in $\{0,1\}^\kappa$. In each round, a node $i \in \mathsf{N}$ can make at most q_i queries to the random oracle H for some non-negative integer q_i. The integer q_i represents the node's "mining power". We denote $Q = \sum_{i \in \mathsf{N}} q_i$ as the total mining power of all nodes in N. The total mining power of all malicious nodes is at most $\rho \cdot Q$ where $\rho \in (0, 1/2)$ is the fraction of adversarial hash power. Further, the adversary can re-distribute the mining powers arbitrarily among the malicious nodes. This generalizes the "flat" model in [21] in which all nodes have the same mining power.

The non-flat mining model allows nodes with zero-mining power in the SNM model. If we consider the flat mining model, the fraction of malicious nodes η will equal the fraction of malicious mining powers ρ. As the adversary cannot control more than 50% of mining power, it cannot control more than 50% of nodes in the network. In the non-flat model, the adversary can control a large number of (sybil) nodes with zero mining power. Thus, the fraction of malicious nodes η can be arbitrarily close to 1.

Point-to-point communication. Any two nodes can establish a new connection or drop an existing connection. A node can only send messages to its neighbors, to whom it has established connections. These messages are guaranteed to be delivered to the neighbors after certain bounded number of rounds.

Establishing/dropping connections. It takes $t_e \in \mathbb{N}$ rounds to establish a new connection between two nodes. At round r, two nodes start establishing a connection with each other. Then, at round $r + t_e$, they can start exchanging messages. Once the connection is established, the nodes can instantly drop it.

Delivering messages. A node u only sends messages to a node v if there is a connection between u and v in that round. The adversary $\mathcal{A}$ is responsible for delivering all messages. The adversary cannot forge or alter the messages of honest nodes, but it can delay or reorder the messages as long as they are delivered within some *bounded delivery delay* $\delta \in \mathbb{N}$ rounds.

Connection observation. For the static adversary, we consider that newly established connections can be instantly observed by the adversary. In the full version we discuss the adaptive adversary and restrict their ability to observe connections. Specifically, the adversary can only observe new connections after a certain amount of time has passed.

2.2 Security Properties

We say a blockchain protocol is secure if it maintains a ledger that satisfies the *persistence* and *liveness* properties. The persistence property states that if a transaction is added to the ledger of an honest player, then it will be added to the same position in the ledgers of all honest players. The liveness property

states that if a transaction is given as input to all honest players, the transaction should eventually appear on the ledgers of honest players. The security analysis [21] of a blockchain protocol relies on a property that any valid message can be disseminated from any honest node within a bounded time. We refer to this property as *reliable dissemination*.

Notations. Let $\mathcal{C}_i^r$ be the local chain of player i at round r. We define the persistence and liveness properties in the joint view $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ as follows. We define $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ to denote that VIEW in the support of $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$. We define a function $\varepsilon(\cdot)$ is a *negligible* such that for any polynomial $\mathsf{poly}(\cdot)$, there exists some $\kappa_0 \in \mathbb{N}$ in which $\forall \kappa \geq \kappa_0, \varepsilon(\kappa) \leq \frac{1}{\mathsf{poly}(\kappa)}$.

A brief admissibility convention. Throughout Section 2 and the security definitions below, we use the term Γ_{stat}-*admissible environment* to mean an execution tuple $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ such that $\Gamma_{\text{stat}}(n, Q, \eta, \rho, \delta) = 1$, where $\mathcal{A}$ and $\mathcal{Z}$ are probabilistic polynomial-time (PPT), the adversary controls at most an η fraction of nodes and at most a ρ fraction of the total mining power Q, honest messages are delivered within bounded delay δ, and the environment only supplies inputs that preserve ledger validity for honest nodes. In the static model considered here, the adversary can also instantly observe newly established connections. A fuller formal statement appears in Appendix A, but this brief convention is the one used by Definitions 1–5 in the main text.

Persistence. For a $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{per}(\text{VIEW}) = 1$ iff for any honest player i at round r and any transaction tx in the ledger $\Lambda(\mathcal{C}_i^r)$, there exists some network delay $\Delta \in \mathbb{N}$ such that any honest player j at round $r' \geq r + \Delta$, we have $\Lambda(\mathcal{C}_i^r, tx) = \Lambda(\mathcal{C}_j^{r'}, tx)$.

Definition 1 (Persistence). *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* persistence in Γ_{stat}-environments *if* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{per}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa)$.

Liveness. For $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, we define $\mathsf{liv}(\text{VIEW}) = 1$ iff there exists a "wait time" parameter $\mathsf{t} = O(\kappa)$ such that, for any transaction tx that is given as input to all honest players from round r to round $r+\mathsf{t}$, we have $\Lambda(tx, \mathcal{C}_i^{r+\mathsf{t}}) > 0$ for every honest player i at round $r + \mathsf{t}$.

Definition 2 (Liveness). *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* liveness in Γ_{stat}-environments *if* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{liv}(\text{VIEW}) = 1] \geq 1 - \varepsilon(\kappa)$.

Reliable dissemination. The *reliable dissemination* property states that when an honest node receives (or generates) a valid message, that message will, with high probability, be disseminated to all honest nodes in the network within a bounded time. For a $\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, consider a message msg that is first known by an honest player i at round r_{msg}. Here, the message msg could either be generated by the honest player i or sent from the adversary. For a parameter $\Delta \in \mathbb{N}$, we define $\mathsf{rel}(\text{VIEW}, \Delta) = 1$ iff for any message msg that is known by an honest node at round r_{msg}, all honest nodes received the message msg at round $r_{msg} + \Delta$.

Definition 3 (Reliable dissemination). *Protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* Δ-reliable dissemination in Γ_{stat}-environments *if* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{rel}(\text{VIEW}, \Delta) = 1] \geq 1 - \varepsilon(\kappa)$.

The security analysis in Pass et al. [21] uses reliable dissemination[5] as an assumption. While this condition holds for fully connected networks, it may not necessarily hold for real-world blockchain networks, e.g., Bitcoin.

2.3 Consensus in Sparse Network

We now introduce the notion of network sparsity and then define the "consensus in sparse network" problem.

Network sparsity. The *sparsity* of a network is measured as *the average number of connections per honest node within a given period.* Consider a blockchain protocol execution, a VIEW in the support of $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, and a parameter $t_{\text{s}} = \Omega(\kappa)$. For a round r, let m_r be the number of connections that are established at some round $r' \in [r, r+t_{\text{s}}]$ and have at least one honest endpoint. Let n_r be the number of nodes that are honest at round $r + t_{\text{s}}$. We define the sparsity as $d_r = \frac{m_r}{n_r}$, the average number of connections per honest node from round r to round $r + t_{\text{s}}$. For parameters d, t_{s}, we define $\mathsf{spa}(\text{VIEW}, d, t_{\text{s}}) = 1$ iff for any round r in the execution VIEW, we have, $d_r \leq d$.

Definition 4 (Network sparsity). *A blockchain protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *achieves* d-sparsity in Γ_{stat}-environments *if there exists a parameter* $t_{\text{s}} = \Omega(\kappa)$ *such that* $\Pr[\text{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa) : \mathsf{spa}(\text{VIEW}, d, t_{\text{s}}) = 1] \geq 1 - \varepsilon(\kappa)$.

Remark. The window length t_{s} is part of the sparsity notion: we measure the average number of connections per honest node over a window of length $t_{\text{s}} = \Omega(\kappa)$ so that sparsity captures long-lived communication structure rather than instantaneous connection churn.

We define the consensus in sparse network as follows.

Definition 5 (Consensus in Sparse Network). *The goal is to construct a consensus protocol* $(\Pi^{\mathsf{V}}, \Lambda)$ *in a* Γ_{stat}*-admissible environment that can achieve together 1)* persistence *and* liveness*; and 2)* sparsity*.*

Definition 6 (Bootstrapping condition). *We say an execution satisfies the* bootstrapping condition *if, at the beginning of epoch* 1*, the honest nodes are connected by an initial overlay* G_0 *such that: (i) every honest node can disseminate a valid message to all honest nodes within* Δ *rounds over* G_0*; (ii) the honest subgraph of* G_0 *has diameter* $O(\log n)$*; and (iii) the first epoch has enough time to disseminate blocks and registrations over* G_0 *before the first CoSpaN-generated topology is activated. This condition can be realized either by a trusted setup or by a one-time decentralized bootstrapping procedure. Operationally, one can think of*

[5] The security analysis in [21] assume a *perfect reliable dissemination*, i.e., the messages from honest nodes will *always* be disseminated to all other honest nodes.

G_0 as arising from short-lived bootstrap peers, DNS seeds, or another one-time membership-discovery mechanism used only before the first CoSpaN-generated epoch begins. The condition is therefore a one-time initialization requirement for epoch 1, not a permanent full-connectivity assumption; after initialization, CoSpaN is responsible for maintaining reliable dissemination using sparse reconfigured overlays.

3 Protocol Design

The CoSpaN protocol extends Nakamoto's protocol [18] and implements a P2P network protocol that provides reliable dissemination. (Additional information is provided in Appendix B.) A node that obtains multiple core registrations in an epoch is represented by multiple logical core identities, and therefore receives proportionally more opportunities to establish connections in the next epoch.

3.1 Protocol design

Protocol Π_{CSN} employs an *epoch-based reconfiguration*, where a new topology is constructed every epoch. The reconfiguration in each epoch is divided into two stages: *core selection* and *topology construction*. The pseudocode of protocol Π_{CSN} can be found in Fig. 1.

Epoch-based Configuration As shown in Fig. 2, each epoch spans L consecutive blocks, where $L \in \mathbb{N}$ is a fixed parameter. For a node, the ith epoch includes rounds with block heights in $[(i-1)L, iL)$, where block height is the length of the longest valid chain observed. Due to network delays and adversarial behavior, block heights may vary across nodes.

Each epoch is divided into two smaller periods: a *registration period* for core selection, followed by a short *grace period* for topology construction. For $k = \Omega(\kappa)$ and $k \ll L$, we set the lengths of the registration and grace periods to $L_{\texttt{reg}} = L - 2k$ and $2k$, respectively. The grace period serves to tolerate any potentially different views of the nodes. During the first half of the grace period, consisting of k blocks, there is sufficient time to rule out any divergence among nodes regarding the set of core registrations, i.e., the set of core nodes in the next epoch. Similarly, the k blocks in the second half allow sufficient time for nodes to establish new connections. During the grace period, the nodes maintain their existing connections and only drop them when the new epoch begins to ensure network connectivity continuity. Why refresh the topology every epoch? Even in the static-adversary setting, periodic refresh serves two purposes. First, mining power can shift over time, and the overlay should continue to track the current resource distribution. Second, re-randomization prevents long-lived structural bias in a sparse network and reduces the security dependence on any single persistent realization of the overlay.

CoSpaN Protocol Π_{CSN}

Parameters:

- Epoch length L, grace period half-length $k = \Omega(\kappa)$
- Core width s, connection parameter d

Local state: Node u has a state $\langle \mathcal{C}, \mathsf{ip} \rangle$ containing blockchain copy $\mathcal{C}$ and its network address ip.

Compute block height $l = len(\mathcal{C})$
Compute epoch number $i = \lfloor l/L \rfloor + 1$
Core selection: If $l < L - 2k$
  Generate a pseudo-identity cid
  Set $\mathsf{context} := \langle \mathsf{prev}, MkRoot(txs), \mathsf{cid} \rangle$
  Upon finding a *nonce* such that
    $H(\mathsf{context}, nonce) \in [\mathsf{T}, \mathsf{T} + \mathsf{T}_{\mathrm{core}}]$
  propagates a core registration
    $\mathsf{cr} = \langle \mathsf{context}, nonce \rangle$.

Topology construction: If $l \in [iL - k, iL)$

Randomness extraction:
If $l = iL - k$, concatenate all the blocks with block heights in $[iL - 3k, iL - 2k)$ and return the hash value as rnd_i

Verifiable random connections:
Let $\bar{\mathsf{C}} = \{$core registrations in the registration period$\}$
Let $\bar{\mathsf{C}}_u \subseteq \bar{\mathsf{C}}$ be the set of core registrations from node u
Core-core connections:
For each core registration $\mathsf{cr} \in \bar{\mathsf{C}}_u$ **do**

- *Step 1: Announcing eligible connections*
  For each core registration $v \in \bar{\mathsf{C}} \setminus \bar{\mathsf{C}}_u$ **do**
    With probability $\bar{\mu}$, announce the address to v via an eligible message.
- *Step 2: Establishing connections*
  For each eligible message from w with the address $w.\mathsf{ip}$
    Connect to $w.\mathsf{ip}$ after verification.
- *Step 3: Accepting connections*
  For each connection request
    Accept the connection after verification.

Core-periphery connections:
[u as a core node] Establishing connections
For each core registration $\mathsf{cr} \in \bar{\mathsf{C}}_u$ **do**
  For each periphery node $v \neq u$ with address $v.\mathsf{ip}$ **do**
    With probability $\bar{\mu}$, connect to $v.\mathsf{ip}$.
[u as a periphery node] Accepting connections
For each connection request from w
  Accept the connection after verification.

Fig. 1: CoSpaN protocol Π_{CSN}.

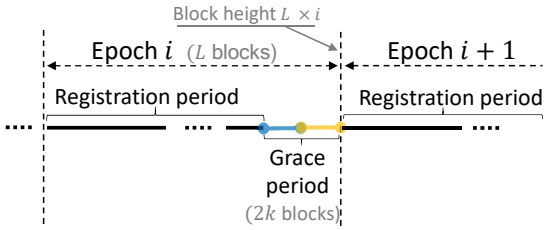


Fig. 2: Epoch of L blocks consisting of 1) a registration period in which nodes use a PoW merged-mining to perform core registration and 2) a short grace period of $2k$ blocks for $k = O(\kappa)$. The first half of the grace period is to guarantee the convergence of nodes' views on the set of core registrations. In the second half of the grace period, nodes will construct a new network topology using verifiable random connections, computed from the core registrations.

Core selection During the registration period, the nodes are selected via the same PoW mining to find new blocks, albeit, with different thresholds [23,1]. Our protocol only requires a miner-proportional, publicly verifiable registration source derived from the same underlying PoW process. A 2-for-1 PoW primitive could be substituted without materially changing the sparse-overlay construction or the subsequent analysis. We present the protocol with merge-mined registrations because this keeps a simple Nakamoto-style interface and aligns directly with the epoch-validity rules used below. Accordingly, we do not rely on full statistical independence between block generation and registration generation beyond what is captured by the protocol-validity rules, the epoch-freshness requirement, and the timing assumptions in the analysis. To generate new blocks, nodes attempt to solve a PoW puzzle by searching for a *nonce* over a $\mathsf{context}$ that satisfies the hash inequality $H(\mathsf{context}, nonce) < \mathtt{T}$, where $\mathtt{T}$ is the mining difficulty. The context $\mathsf{context} = \langle \mathsf{prev}, MkRoot(txs), \mathsf{cid} \rangle$ consists of a pointer (hash value) prev to the previous block, the Merkle root $MkRoot(txs)$, a single hash value to validate the included transactions, and a network pseudo-identity cid. Whenever the node finds a *nonce* that $\mathtt{T} \leq H(\mathsf{context}, nonce) < \mathtt{T} + \mathtt{T}_{\mathrm{core}}$, where $\mathtt{T}_{\mathrm{core}}$ denotes *core registration difficulty*, it generates and propagates a core registration $\mathsf{cr} = \langle \mathsf{context}, nonce \rangle$. The core registrations will be propagated within the P2P network and included in the ledger as (prioritized) transactions. Operationally, this rule is intended to exclude the simplest cross-epoch replay attack directly at the protocol-validity layer. The point is not to claim that merge-mining yields perfect independence between all protocol objects, but to make explicit which information must already be fixed before the next overlay is sampled and which registrations are admissible for that epoch.

Epoch-valid registration rule. Honest nodes treat a registration as valid for epoch i only if it is observed and confirmed within epoch i's registration window and is bound to that epoch through the chain context used to validate it. Registrations replayed from earlier epochs are discarded for topology construction rather than re-used. This freshness rule prevents an adversary from stockpiling registrations from previous epochs and strategically releasing them later. Denote by s the *core width*, which represents the expected number of core registrations (i.e. the

number of core nodes) in each epoch. To ensure $\Theta(s)$ core registrations in each period, we set $\mathtt{T}_{\text{core}} = \frac{s}{L_{\mathtt{reg}}}\mathtt{T}$. This can be verified by noting that the mining difficulty T leads to approximately $L_{\mathtt{reg}}$ blocks during the registration period. Our later security analysis remains valid when the number of actual registrations is within a constant ω, such as $\omega = 90\%$, of the core width s. This adds robustness to the protocol, against low participation of nodes in the core selection.

Core-periphery topology construction During the second half of the grace period, the *core-periphery topology* is constructed in two phases. First, all nodes in the network *extract randomness* from the previous epoch. Then, based on this randomness, core nodes establish *verifiable random connections* to other core and periphery nodes. The probability that a core node establishes a connection to another (core or periphery) node is $\bar{\mu} = \frac{d}{2s}$, where d is the *connection parameter* and s is the *core width*. Note that the expected number of (logical) core nodes associated with the same (physical) node (or miner) i is proportional to its mining power q_i, as is the expected number of connections established by i.

Randomness extraction. Similar to [4], all nodes compute rnd_i as the hash of the concatenation of the last k blocks in the registration period, namely those with block heights in $[iL-3k, iL-2k)$. This ensures that all nodes derive the same rnd_i in the second half of the grace period. We interpret rnd_i as a public beacon fixed before the corresponding VRF-eligibility checks are evaluated, so honest parties commit to their registrations and keys before using rnd_i for connection formation. The intended protection is specific: together with the epoch-valid registration rule, this interpretation rules out cross-epoch replay of old registrations and post-beacon selection of fresh registrations or keys for the next overlay. We do not claim that the protocol thereby eliminates every conceivable grinding strategy in an unrestricted network; rather, the point is that the analysis relies only on excluding those replay-and-selection behaviors that would otherwise invalidate the epoch-by-epoch sampling argument.

Connection establishment. During the second half of the grace period, all nodes utilize core registrations and extracted randomness to establish new connections. The connection establishment procedure is divided into two sub-procedures for two types of connections: *core-core connections* and *core-periphery connections*.

- Core-core connections. Each core node u establishes a connection with every other core node v with a probability of $\bar{\mu}$. Two core nodes u and v are connected if either u establishes a connection to v or v establishes a connection to u.
- Core-periphery connections. Each core node u establishes a connection with every periphery node v with a probability $\bar{\mu}$. Periphery nodes do not establish connections with other nodes.

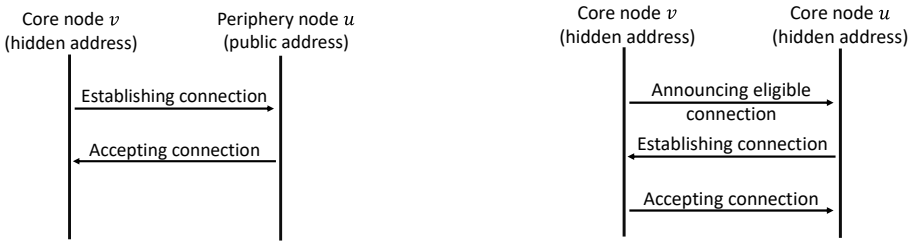
3.2 Security improvements

We describe how nodes establish verifiable random connections to construct the core-periphery graph while preserving core anonymity.

Verifiable random connections To establish verifiable random connections, a public-key pseudorandom function known as a VRF is employed [7]. The VRF provides proofs that its outputs were calculated correctly and randomly, making them difficult to predict. The owner of the secret key can compute the function value σ and an associated proof π for a given input value. Others can use an algorithm called Verify, along with the proof and the associated public key, to check if the function value was calculated correctly without revealing any information about the secret key.

Consider a core node u that holds the key pair $(\mathsf{sk}', \mathsf{pk}')$ for the VRF. For a (core or periphery) node v, let id_v be the identity of node v. Here, if v is a core node, we set the identity id_v as the pseudo-identity cid_v of v. If v is a periphery node, we set the identity id_v as the address ip_v of v. The node u computes $(\sigma_v, \pi_v) := \mathsf{Prove}_{\mathsf{sk}'}(\mathsf{rnd}_i || \mathsf{id}_v)$. We say that u is *eligible* to connect to v, in epoch $i+1$, if and only if output σ_v is smaller than a threshold $\mathtt{T}_{\mathrm{con}} = \bar{\mu} \cdot 2^{\kappa}$, i.e., $\sigma_v < \mathtt{T}_{\mathrm{con}}$. For each node v, the probability that $\sigma_v < \mathtt{T}_{\mathrm{con}}$ is satisfied and node u is eligible to connect to node v is $\bar{\mu}$. Upon receiving a connection request from u, the node v can verify node u is eligible by verifying 1) (σ_v, π_v) is computed correctly, and 2) $\sigma_v < \mathtt{T}_{\mathrm{con}}$.

Restricted exposure of core nodes As discussed above, CoSpaN limits broad exposure of the mapping between logical core identities and physical addresses. The node's address is therefore not included in the pseudo-identity. The relevant security goal is not absolute anonymity of core nodes, but preventing public protocol state from directly revealing a globally visible list of highly connected endpoints. We stress, however, that CoSpaN does *not* claim that a core node's address is hidden from every party that may ever interact with it; rather, adversarial counterparts may still learn the addresses required for the specific connections they form. To generate cid, a node generates key pairs $(\mathsf{sk}', \mathsf{pk}')$, $(\mathsf{sk}, \mathsf{pk})$, $(\bar{\mathsf{sk}}, \bar{\mathsf{pk}})$ for a VRF, a public-key encryption scheme, and a signature scheme, respectively. The network pseudo-identity is returned as $\mathsf{cid} = \langle \mathsf{pk}', \mathsf{pk}, \bar{\mathsf{pk}} \rangle$.



(a) Core-periphery connection. (b) Core-core connection.

Fig. 3: Establishing connections. To preserve core anonymity, an extra step is added to establish core-core connections.

In Fig. 3, we show the interactions to establish core-periphery and core-core connections. The establishment of a core-periphery connection consists of two steps. First, a core node u requests to connect to a periphery node v if it is eligible to do so. Then, node v will accept the connection after verifying that u is eligible. For the core-core connections, because the addresses of core nodes cannot be obtained from the pseudo-identities, an extra step is needed. Each core node u broadcasts an encryption of its address to the core nodes to which it is eligible to connect. More precisely, node u creates a message M containing the relevant eligibility information and broadcasts it to all nodes in the P2P network. For each "eligible" core node v, u uses v's public key, $v.\mathsf{pk}$, for encryption, computes $\xi \leftarrow \mathsf{Enc}_{v.\mathsf{pk}}(\sigma_{\text{core}}, \pi_{\text{core}}, \mathsf{ip})$, and adds ξ to the list M. Node u then computes a signature $\gamma = \textsc{Sign}_{\bar{\mathsf{sk}}}(M, \mathsf{cr})$ and broadcasts (M, cr, γ) to all nodes in the network. By using encryption, u limits the visibility of its network address $u.\mathsf{ip}$ to only those nodes that it is eligible to connect to in the future. To prevent spamming attacks, every node forwards at most one valid message (M, cr, γ) for each core registration cr. The core node v can then decrypt the message to recover node u's address. After verifying that u is eligible, node v requests a connection to node u by sending a signature $\textsc{Sign}_{\bar{\mathsf{sk}}}(v.\mathsf{cr})$. Node u accepts the connection if node v's signature is correct.

This mechanism limits broad exposure of core addresses but does not eliminate all leakage channels. Adversarial endpoints may still learn the addresses of the specific core nodes to which they connect, and side channels such as traffic analysis, routing visibility, active probing, or timing inference lie outside our formal model. Accordingly, the slow-observation model should not be read as asserting that delayed observability is realistic in every open P2P deployment. Its role is narrower: to characterize the guarantees available when newly established honest-honest links are exposed only after enough operational delay — for example because of indirection, shielding, or other friction — to matter relative to the overlay refresh schedule. The DoS-resilience claim should therefore be read as making broad targeting from public information substantially harder, not as preventing every possible leakage path.

Overhead. In each epoch, a core node performs $O(n + d \cdot s)$ work: it evaluates VRFs for candidate peers, encrypts eligibility information for the core nodes it can contact, and decrypts messages received from other core nodes. A periphery node performs $O(d)$ verification work. The communication cost per epoch is $O(n{\cdot}s{+}n{\cdot}d{+}s{\cdot}d)$: the $O(n{\cdot}s)$ term comes from disseminating encrypted eligibility messages, and the remaining terms come from connection establishment and use of the sparse overlay. On chain, the protocol adds one record per successful registration per epoch, so storage is governed by the core-width parameter s and the epoch length L, not by all pairwise connections in the overlay.

4 Security analysis

In Table 2, we summarize the security analysis of the CoSpaN protocol against a static adversary and two additional weakly adaptive adversaries. An M-adaptive

adversary needs to *wait for τ_{corr} rounds before corrupting an honest node*. An S-adaptive adversary can corrupt a node instantly, but must *wait for τ_{obs} rounds before discovering a newly established connection*. We consider an S-adaptive$\langle\phi, \gamma\rangle$ setting, in which the top ϕ fraction of honest nodes control a fraction γ of mining power. The S-adaptive setting can be seen as a special case of S-adaptive$\langle\phi, \gamma\rangle$ with $\phi = 1$ and $\gamma = 1 - \rho$.

These adaptive results should be read as guarantees against delayed-reaction adversaries. The M-adaptive model assumes nontrivial delay before a chosen node can be corrupted, while the S-adaptive model assumes delayed observability of newly established honest-honest connections. We therefore present the adaptive theorems as conditional guarantees under explicit timing assumptions rather than as claims about the strongest fully adaptive open-network adversary. In particular, the S-adaptive model is not intended to capture an attacker with immediate traffic visibility. The theorems are best interpreted for deployments in which operational friction or an auxiliary shielding layer slows node compromise or connection exposure, and their purpose is to characterize what sparse PoW overlays can guarantee under such reaction-time assumptions. These hypotheses do real work: the static analysis already requires epoch lengths of order $\Omega(\kappa)$, and the adaptive statements additionally require the relevant delay parameter to be large enough that such epochs fit inside the assumed reaction window. We therefore do not present the adaptive results as saying that the required delays are mild in every deployment; rather, they identify the exact regimes in which rigorous guarantees remain provable despite the sparse-network impossibility intuition for fully reactive attackers.

Threat model	Adaptive corruption	Connection observability	Epoch length L	Core width s	Connection parameter d
Static	n/a	Instantly	$\Omega\left(\frac{\kappa}{(1-\rho)\omega}\right)$	$\Omega\left(\frac{\kappa}{1-\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-\rho)\omega)} + \log s\right)$
M-adaptive	τ_{corr} rounds	Instantly	$O\left(\tau_{\text{corr}} \cdot \alpha\right)$	$\Omega\left(\frac{\kappa}{1-\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-\rho)\omega)} + \log s\right)$
S-adaptive	Instantly	τ_{obs} rounds	$O\left(\tau_{\text{obs}} \cdot \alpha\right)$	$\Omega\left(\frac{\kappa+h}{1-2\rho}\right)$	$O\left(\frac{\kappa}{-\log((1-2\rho)\omega)} + \log s\right)$
S-adaptive$\langle\phi, \gamma\rangle$	Instantly	τ_{obs} rounds	$O\left(\tau_{\text{obs}} \cdot \alpha\right)$	$\Omega\left(\frac{\kappa+h\cdot\phi}{\gamma-\rho}\right)$	$O\left(\frac{\kappa}{-\log((\gamma-\rho)\omega)} + \log s\right)$

Table 2: Feasible parameter settings of CoSpaN (L, s, d) against different adversary models. CoSpaN, parameterized by the epoch length L, the core width s, and the connection parameter d can achieve reliable dissemination with a sparsity of $\frac{2}{1-\eta}(1 + \frac{s}{n})d = O(\log n)$, where n is the number of nodes and η denotes the fraction of malicious nodes. Defending against (weakly) adaptive adversaries requires (1) shorter epoch lengths and (2) larger numbers of core nodes to limit the effect of targeted corruption toward the core nodes. In an S-adaptive$\langle\phi, \gamma\rangle$ model, where $\phi \in (0, 1)$ and $\gamma \in (\rho, 1 - \rho)$), we consider an S-adaptive adversary when the top ϕ fraction of honest nodes control a fraction γ of mining power. Accordingly, the adaptive rows in this table should be read as feasible parameter regimes under explicit delay assumptions, rather than as deployment claims for arbitrary open-network adversaries.

Main theorem We prove that by choosing sufficient parameters the CoSpaN protocol can achieve security properties (defined in Section 2.2) under the presence of a static adversary in a network with $O(\log n)$ sparsity.

Theorem 1 (Static security in sparse network). *Consider Γ^*_{stat}-admissible environments and protocol Π_{CSN} with parameters $L > \frac{k}{(1-\rho)\omega} + 2k$, $s > k$ and $d \geq \max(\frac{k}{-\log \mathfrak{q}}, 4 \log s)$. Under the bootstrapping condition of Definition 6, protocol Π_{CSN} achieves persistence, liveness, and D-sparsity, where $D = 2(1 + \epsilon)(1 + \frac{s}{n})/(1 - \eta)d$.*

Additional information is provided in Appendix C. Please note that the security analysis in this section is highly non-trivial; due to space limitations, lots of results in this section have to be omitted from this submission; please refer to the full version for the proof details.

5 Numerical Studies

We compare CoSpaN with two Bitcoin-like baselines by measuring the adversary's cost to break dissemination and to mount double-spending attacks. We also report the cost of defending against the adaptive models and the effect of mining-power concentration; additional plots appear in Appendix D.

5.1 Setup

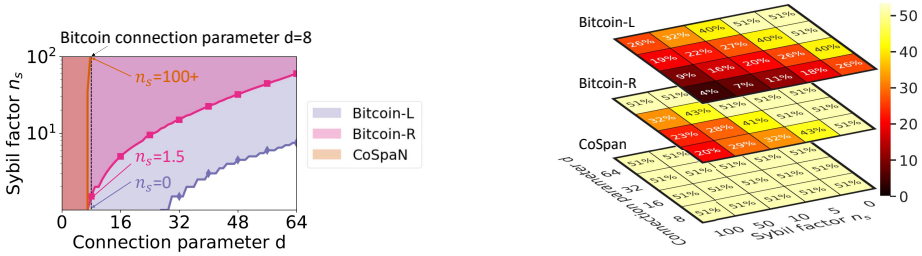
We compare CoSpaN with two baselines of similar sparsity (average degree d). **Bitcoin-L(ike).** Each node opens $d = 8$ outbound links and accepts up to $d_{max} = 125 \approx 15d$ inbound links, following the randomized-connection defenses of [12]. We conservatively allow each adversarial node to request links from all honest nodes. **Bitcoin-R(andom).** Each unordered node pair is connected with probability d/n, yielding expected degree d and no inbound cap; Sybil nodes therefore increase adversarial incidence only through their number.

Parameter settings. Unless noted otherwise, we use 1,000 honest nodes, 1,000 malicious nodes, and $2{,}000 \cdot n_s$ additional zero-mining-power Sybils, so $\eta = 1/(2(1 + n_s))$. The adversary controls mining fraction $\rho = 0.3$, the participation rate is $\omega = 0.9$, and CoSpaN uses core width $s = 0.05n$. These defaults stress-test sparse dissemination under strong but sub-majority adversaries: $\rho = 0.3$ is substantial but below the honest-majority threshold, $\omega = 0.9$ allows occasional non-participation, and $s = 5\%$ keeps the backbone sparse while maintaining a sufficiently large honest core. This 5% choice is a representative operating point rather than a uniquely required value.

Metrics. We measure (i) the *Sybil factor* needed to disconnect the network and (ii) the *adversarial mining power* needed for a double-spending attack. We also study the honest mining power required by CoSpaN in different settings and the relation between degree and mining power. All reported values are averages over 1,000 runs.

5.2 Costs of attacks

We focus on static settings.



(a) Sybil resilience. (b) Double-spending threshold.

Fig. 4: At Bitcoin's default sparsity $d = 8$, CoSpaN remains connected until the adversary creates an unrealistically large number of Sybil nodes ($n_s > 100$), while Bitcoin-L and Bitcoin-R fail at $n_s = 0$ and $n_s \approx 1.5$, respectively. Under $n_s = 10$ and $d = 8$, CoSpaN still requires the adversary to reach 51% mining power for a 12-confirmation double spend, whereas Bitcoin-L and Bitcoin-R become vulnerable at much smaller mining shares after network splitting.

Sybil attacks. For a static adversary with 30% mining power, Fig. 4a shows that the cost of breaking reliable dissemination in CoSpaN is much higher than in the baselines. At Bitcoin's sparsity level $d = 8$, the adversary needs $n_s > 100$ to disconnect CoSpaN, versus $n_s = 0$ for Bitcoin-L and $n_s \approx 1.5$ for Bitcoin-R; even at $d = 64$, both baselines can still be broken with fewer than 100 Sybils.
Double-spending attacks. For a static adversary with $n_s = 10$ and $d = 8$, Fig. 4b studies the mining power needed to reverse a 12-confirmation transaction with probability above 10% using network splitting plus block withholding [21]. Because CoSpaN keeps the honest network connected, the adversary still needs 51% mining power. In contrast, splitting Bitcoin-L and Bitcoin-R reduces the relevant honest component, so much smaller mining shares suffice; at $n_s = 100$ and $d = 8$, the required mining power drops to 4% and 20%, respectively.
Acknowledgement. Thang N. Dinh's work was partially supported by NSF AMPS award 2229075, NSF SaTC award 2140411, and Commonwealth Cyber Initiative (CCI) awards. Hong-Sheng Zhou's work was partially supported by NSF grant CNS-1801470, and the VCU Quest Fund.

References

1. Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia C. Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. *ACM CCS 2019*, pages 585–602.
2. Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. *ACM CCS 2014*, pages 15–29, 2014.

3. Sandro Coretti, Aggelos Kiayias, Cristopher Moore, and Alexander Russell. The generals' scuttlebutt: Byzantine-resilient gossip protocols. *ACM CCS 2022*, pages 595–608.
4. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98.
5. Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10.
6. John Dillon. Bitcoin-development mailinglist: Protecting bitcoin against network-wide dos attack, 2018.
7. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. *PKC 2005*, volume 3386 of *LNCS*, pages 416–431.
8. Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.
9. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310.
10. Juan A. Garay, Aggelos Kiayias, and Yu Shen. Permissionless clock synchronization with public setup. *TCC 2022, Part III*, volume 13749 of *LNCS*, pages 181–211.
11. Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *ACM CCS 2016*, pages 3–16.
12. Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. *USENIX Security 2015*, pages 129–144.
13. Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{O}(n^{3/2})$ bits. In *DISC 2009*, pages 464–478.
14. Chen-Da Liu-Zhang, Christian Matt, Ueli Maurer, Guilherme Rito, and Søren Eller Thomsen. Practical provably secure flooding for blockchains. *ASIACRYPT 2022, Part I*, pages 774–805.
15. Chen-Da Liu-Zhang, Christian Matt, and Søren Eller Thomsen. Asymptotically optimal message dissemination with applications to blockchains. Cryptology ePrint Archive, Report 2022/1723, 2022.
16. Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. *IACR Cryptol. ePrint Arch.*, 2018:236, 2018.
17. Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 400–430.
18. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.
19. Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. Erlay: Efficient transaction relay for Bitcoin. *ACM CCS 2019*, pages 817–831.
20. Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *EuroS&P 2016*, pages 305–320.
21. Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673.
22. Rafael Pass and Elaine Shi. FruitChains: A fair blockchain. *ACM PODC 2019*, pages 315–324.

23. Peter R Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.
24. Ye Wang and Roger Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 178–188, 2020.
25. Karl Wüst and Arthur Gervais. Ethereum eclipse attacks. Technical report, ETH Zurich, 2016.

A Supplemental Material for Section 2

A blockchain execution. The execution of the blockchain protocol is directed by the environment $\mathcal{Z}$, which initiates the set of nodes N. The nodes are initially designed as honest or malicious. We consider the execution with a polynomial in κ number of rounds. In each round, each node receives a list of transactions txs as input from the environment and adjusts its connections according to Π. Each honest node executes Π and incorporates any input and messages from its neighbors into its local data structure $\mathcal{C}$. Each node $i \in \mathsf{N}$ accesses the random oracle $H(\cdot)$ exactly q_i times with different nonces to find a valid proof of work (PoW). If an oracle call returns a proof of work, the node can generate a new block and send the new block to its neighbors, who will propagate it further.
Static adversary. We start with a static adversary $\mathcal{A}$ that controls the same subset of nodes throughout the entire execution of the protocol. In the full version, we will analyze the security of our protocols against *(weakly) adaptive adversaries* that can corrupt more nodes during the protocol execution, but in certain restricted manners. To enable security in the presence of weakly adaptive adversaries, we will consider a restricted point-to-point communication model in which the adversary can only observe the connections among honest nodes after a sufficiently long time. For adversary $\mathcal{A}$, and environment $\mathcal{Z}$, let $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$ be a random variable denoting the joint view of all nodes (i.e., all their inputs, randomness, and messages received) over all rounds.

Admissible environments. We will be considering executions with restricted adversaries and environments. Given a predicate $\Gamma_{\text{stat}}(\cdot,\cdot,\cdot,\cdot,\cdot)$, we define Γ_{stat}-admissible environment as follows.

Definition 7 (Γ_{stat}-admissible environments). *A tuple $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ with $\Gamma_{\text{stat}}(n, Q, \eta, \rho, \delta) = 1$ is Γ_{stat}-admissible w.r.t $(\Pi^{\mathsf{V}}, \Lambda)$ if $\mathcal{A}$, $\mathcal{Z}$ are* PPT *algorithms, and for every* VIEW *in the support of $\mathsf{EXEC}_{\Pi^{\mathsf{V}},\mathcal{A},\mathcal{Z}}(\kappa)$, the following holds:*

- *$\mathcal{A}$ controls at most $\eta \cdot n$ nodes, where n is the total number of nodes. The total mining power controlled by $\mathcal{A}$ is at most ρQ, where Q is the total mining power.*
- *In every round r, the environment $\mathcal{Z}$ only sends the list of transactions txs as input to an honest player that maintains a ledger $\mathcal{L} = \Lambda(\mathcal{C})$ if the list of transactions can be appended to the ledger, i.e., $\mathsf{V}(\mathcal{L} \| txs) = 1$.*
- *A message sent by an honest node is delayed by at most δ rounds before sending to its neighbors.*

– *$\mathcal{A}$ can instantly observe newly established connections among nodes.*

When context is clear, we refer $(n, Q, \eta, \rho, \delta, \mathcal{A}, \mathcal{Z})$ as Γ_{stat}-admissible.

B Supplemental Material for Section 3

Computation cost. On average, each core node performs 1) $n + s$ times to compute the VRF (each for a core or periphery node); 2) d times to encrypt (each for an eligible core node); and 3) ds times to decrypt (for every encryption from core nodes). In total, the time complexity of a core node in each epoch is $O(n + d \cdot s)$. On average, each periphery node performs d times to verify the connection. Thus, the time complexity of a periphery node in each epoch is $O(d)$.

Communication cost. In an epoch, each core node broadcasts a message that contains the encryption to all other node. The message complexity to broadcast the encryption is $O(n \cdot s)$. The remaining communication happens between the nodes that are connected to each other in that epoch. The message complexity to this communication is $O(n+s)d$. Thus, the total message complexity in each epoch is $O(n \cdot s + n \cdot d + s \cdot d)$.

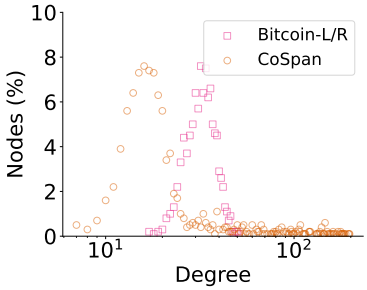
C Supplemental Material for Section 4

Additional security statements. In the full version we in addition show that the CoSpaN protocol can also achieve security properties under the presence of weakly adaptive adversaries.

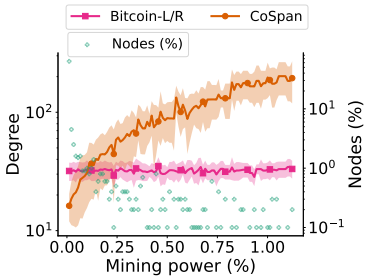
Theorem 2 (M-adaptive security). *Let $k = \Omega(\kappa)$ and consider $\Gamma^*_{\text{M-adap}}$-admissible environments and protocol Π_{CSN} with parameters $\frac{k}{(1-\rho)\omega} + 2k < L < \frac{\tau_{\text{corr}}(1-\epsilon)\alpha}{2}$, $\epsilon \in (0,1)$, $s > k$ and $d \geq \max(\frac{k}{-\log \mathsf{q}}, 4 \log s)$. Under the bootstrapping condition of Definition 6, protocol Π_{CSN} achieves persistence, liveness, and sparsity.*

Theorem 3 (S-adaptive security). *Consider $\Gamma^*_{\text{S-adap}}$-admissible environments and protocol Π_{CSN} with parameter $s > k + \frac{2(1+\epsilon)}{\mathsf{q}_{\text{M-adap}} \cdot \epsilon^2 \log_2 e} h$, $\frac{k}{(1-2\rho)\omega} + 2k < L < \frac{\tau_{\text{obs}}(1-\epsilon)\alpha}{2}$, and $d \geq \max(\frac{k}{-\log \mathsf{q}_{\text{M-adap}}}, 4 \log s)$. Under the bootstrapping condition of Definition 6, protocol Π_{CSN} achieves persistence, liveness, and sparsity.*

Theorem 4 (S-adaptive $\langle \phi, \gamma \rangle$). *Consider Γ^*_{conc}-admissible environments and the protocol Π_{CSN} with parameters $\frac{k}{(\gamma-\rho)\omega} + 2k < L < \frac{\tau_{\text{obs}}(1-\epsilon)\alpha}{2}$, $s > k + h\phi$ and $d \geq \max(\frac{k}{-\log \mathsf{q}_{\text{conc}}}, 4 \log s)$. Under the bootstrapping condition of Definition 6, protocol Π_{CSN} achieves persistence, liveness, and sparsity.*



(a) Degree distribution distribution.



(b) The correlation between mining power and degree of nodes. The shaded areas show the 99% confidence intervals.

Fig. 5: The correlation between the mining power and the degree of nodes in CoSpaN and Bitcoin-random network. The distribution of Bitcoin-L and Bitcoin-R are the same.

D Supplemental Material for Section 5

D.1 Network characteristics

Consider a network in which the mining power of nodes follows a power law distribution. We analyze the degree distribution and the relation between the degree distribution and the mining power distribution of the honest nodes. From Fig. 5a, the degree distribution of Bitcoin-L and Bitcoin-R networks follows a normal distribution. The degree distribution of the CoSpaN network follows a power law distribution. Specifically, as shown in Fig. 5, the degrees of nodes in Bitcoin-L and Bitcoin-R networks are independent of the mining power. The degree of nodes in those networks has the same distribution. Meanwhile, in the CoSpaN network, the degree of nodes is proportional to the mining power.

D.2 Further notes on the default numerical parameters

The default setting in Section 5.1 is intended as a stress test rather than a tuned best case. We use 1,000 honest nodes and 1,000 malicious nodes, then add $2{,}000 \cdot n_s$ extra zero-mining-power Sybils to sweep the Sybil factor over a wide range. The mining-power parameter $\rho = 0.3$ models a strong but sub-majority adversary, $\omega = 0.9$ allows occasional missed registrations, and $s = 0.05n$ is a representative sparse operating point large enough to keep the honest logical core well populated while still maintaining a sparse backbone. We do not claim that 5% is uniquely optimal; it is a convenient baseline for comparing sparsity, attack cost, and overhead without moving to an unusually dense regime. The figures should therefore be read as comparative stress tests at matched or comparable sparsity, not as deployment-specific forecasts.