Security-Performance Tradeoff in DAG-based Proof-of-Work Blockchain Protocols

Shichen Wu
Shandong University
shichenw@mail.sdu.edu.cn

Puwen Wei
Shandong University
pwei@sdu.edu.cn

Ren Zhang
Cryptape Vanguard and Nervos
ren@nervos.org

Bowen Jiang
Shandong University
bowen.jiang@mail.sdu.edu.cn

***Abstract*—Proof-of-work (PoW) blockchain protocols based on directed acyclic graphs (DAGs) have demonstrated superior transaction confirmation performance compared to their chain-based predecessors. However, it is uncertain whether their security deteriorates in high-throughput settings similar to their predecessors, because their acceptance of simultaneous blocks and complex block dependencies presents challenges for rigorous security analysis.**

We address these challenges by analyzing DAG-based protocols via a congestible blockchain model (CBM), a general model that allows case-by-case upper bounds on the block propagation delay, rather than a uniform upper bound as in most previous analyses. CBM allows us to capture two key phenomena of high-throughput settings: (1) simultaneous blocks increase each other's propagation delay, and (2) a block can be processed only after receiving all the blocks it refers to. We further devise a reasonable adversarial block propagation strategy in CBM, called the late-predecessor attack, which exploits block dependencies to delay the processing of honest blocks. We then evaluate the security and performance of Prism and OHIE, two DAG-based protocols that aim to break the security-performance tradeoff, in the presence of an attacker capable of launching the late predecessor attack. Our results show that these protocols suffer from reduced security and extended latency in high-throughput settings similar to their chain-based predecessors.

I. Introduction

Nakamoto consensus (NC), implemented in Bitcoin [27] and hundreds of subsequent digital currencies [25], is the first protocol to maintain an inalterable ledger without relying on any prior knowledge of the participants' identities. In NC, the ledger is organized as a chain of *blocks*, each containing a set of confirmed transactions. To extend this *blockchain*, NC's participants, known as *miners*, compete to solve a cryptographic puzzle generated from a group of new transactions and the latest block. Finding a solution to this puzzle allows its miner to broadcast a block packing these transactions. This process is called *proof-of-work (PoW)*. Miners who accept the block update their ledgers and start working on the next puzzle.

Although NC is widely recognized as a technical breakthrough, its poor performance—low throughput and high transaction confirmation latency—prevents it from processing transactions on a global level. These limitations are rooted in NC's security demands, which require that most blocks be mined after the majority of miners have received the blocks' predecessors [12], [29], [38]. This requirement can only be guaranteed by upper-bounding the block size and lower-bounding the block interval. These bounds prevent NC from reaching the networks' *physical limits* of throughput and latency [4], which are the throughput reaching the network's capacity and the latency proportional to the transaction propagation delay.

A popular approach to breaking the security-performance tradeoff and reaching the physical limits is the DAG-based protocols. These protocols allow a block to refer to multiple predecessor blocks, thus replacing the single-chain-based ledger structure with a directed acyclic graph (DAG). As simultaneous blocks can all contribute to transaction confirmation, these protocols [4], [5], [20], [21], [36], [37], [44] loosen the limit on the block interval and thus outperform NC in their throughput. However, it is uncertain whether most of them can maintain the same—if not stronger—level of security as NC, because they only offer weak security guarantees or even flawed security analyses (Sect. II). This situation renders it difficult to evaluate how security and performance interact with each other. The only two exceptions are Prism [4] and OHIE [44], who, in addition to their outstanding performance, prove their security following NC's properties. Concretely, both designs prove that they can tolerate an adversarial mining power share of close to 50%—same as NC, and this threshold—unlike NC's—is (almost) independent of the throughput. These encouraging results make us wonder: *are Prism and OHIE completely exempt from the security-performance tradeoff, even when executing at the physical limits?*

In this paper, we show that this is not the case. We observe that both designs implicitly rely on the following key assumption in their proofs to decouple security and performance:

Assumption of Decoupling. *If some types of blocks are small enough and enjoy a* **priority propagation policy**, *i.e., they are propagated before all other blocks in network congestion, then (1) they can be propagated within a fixed and short network propagation delay D, and (2) all miners can* **accept these blocks**, *i.e., include them in their working puzzle, immediately after receiving them.*

These special blocks, which we call *priority blocks*, are then used at the core of their security proofs to bypass the influence of network congestion thanks to their short and constant delay. However, we argue that when executing at the physical limits, two inevitable phenomena would invalidate this assumption:

Phenomenon 1 (Block Jam)**.** *If in a given time interval t, the total size of the newly mined priority blocks exceeds $C \cdot t$, where C is the network's processing capacity, then not all of these blocks can be propagated at D, violating outcome (1) in the assumption.*

A blockchain may suffer from block jams even if it does not operate at its physical limits. For example, the bandwidth may be temporarily occupied by other applications.

Phenomenon 2 (Late Predecessor)**.** *When a priority block finishes propagation within D, if one of its predecessors (not necessarily a priority block) has not finished propagation, miners who have not received this predecessor cannot accept and work on the priority block, violating outcome (2).*

By violating the Assumption of Decoupling, these phenomena reveal that DAG-based protocols are not immune to the security-performance tradeoff. These phenomena thus uncover a gap between these protocols' actual performance limits/potential and their theoretical analyses. Unfortunately, there is no easy solution to this issue, whether in practice by modifying designs or in theory by patching analyses. In practice, these phenomena of "overlapping block propagation" cannot be eliminated from DAG-based protocols as their performance gain over NC depends on these simultaneous blocks. In theory, we cannot "sweep the problem under the rug" by using a larger D to bound the extended propagation delay, because this quick hack would not only disregard the intricacies of network-layer attacks [26], [42] and the protocols' priority propagation policies, but also, as we will demonstrate, worsen the protocols' security and performance claims to those of NC. This situation thus calls for a new block propagation model for DAG-based protocols that can quantify their performance gain over NC without overlooking the security-performance tradeoff.

This paper answers this call by proposing a *congestible blockchain model (CBM)*, which captures both phenomena by allowing case-by-case upper bounds on the propagation delay. CBM is a general yet simple model that encapsulates the key parameters affecting the block propagation process, including the block size, the block generation frequency, the blockchain topology, the priority propagation policies, and the network's physical limits. We then explore how an attacker can tamper with the blocks' propagation and acceptance in CBM, leading to a reasonable attack strategy called the *late-predecessor attack (LP attack)*. By focusing on the network layer and remaining agnostic about the consensus rules, this attack is applicable to all DAG-based protocols. Lastly, we update both the security analyses and the simulations of Prism and OHIE against an attacker who can launch this attack, thus quantifying how they are affected by the security-performance tradeoff. Our main challenges and contributions include:

Modeling Block Propagation and Acceptance in DAG-based Protocols. Existing formal analyses of NC [10], [12]–[15], [19], [29], [35], [38], [42], [46] adopt a similar block propagation and acceptance model, which we call the *uniform-delay blockchain model (UDBM)*. In UDBM, (1) the upper bound of the block propagation delay is fixed and uniform; (2) as long as this upper bound is reached, a valid block, along with all its ancestors, is accepted by all receivers; (3) all but one block that overlap with each other's propagation are invalid. None of these arguments hold in DAG-based protocols, indicating that the latter demands more delicate modeling.

CBM addresses these challenges by making two changes to UDBM. First, we abandon the uniform delay and instead allow the environment to determine a block's maximum delay when it is generated, which generalizes (1) and thus covers block jams. Note that the maximum delay must be chosen from a finite set to ensure that CBM is synchronous. Second, in CBM, a miner receiving a block does not imply that all its predecessors have been received and accepted by the miner, which corrects (2) and relaxes (3), and thus covers late predecessors. Consequently, CBM incorporates the delicacies of DAG-based protocols' block propagation and acceptance.

Designing the Late-Predecessor Attack. As the core instrument to quantify the security-performance tradeoff, we want the LP attack to (1) capture the effects of the two phenomena, yet still (2) stay close to realistic attacks to maximize its practicality, and (3) be applicable to all DAG-based PoW protocols. Inspired by several reasonable heuristics, we focus on the network layer and consider a new attacker goal: maximizing the (priority) blocks' average *actual delay*, i.e., the interval between a block's generation and the arrival of all its predecessors for some node. In other words, the attacker tampers with all messages' propagation within CBM's constraints to maximize the quantity and the extent of late predecessors. Locating such a strategy is challenging, as the attacker must balance the proportions of mining power that (1) he hopes to mine a late predecessor and a successor block that refers to it, and (2) he hopes to be affected by the reversed arrival sequence, without knowing *a priori* when or by whom these blocks are mined. The resulting strategy establishes a concrete relation between a protocol's key parameters and the block propagation process, which can be employed to analyze the security properties of blockchain under CBM.

Analyzing the Security-Performance Tradeoff of Prism and OHIE. We then analyze the security and performance of Prism and OHIE, in the presence of an attacker capable of launching the LP attack. Specifically, we choose their priority blocks as the attack targets, and other blocks as potential late predecessors. This differentiation allows us to extend their proofs by replacing the short and uniform propagation delay—due to their Assumption of Decoupling—with the detailed block propagation and acceptance process under attack. These techniques of incorporating the network propagation process into proving blockchain security properties are of independent interest. The results in Fig. 5 and Fig. 6 show that although both protocols outperform NC, they are not exempt from the security-performance tradeoff.

Our Prism analysis reveals that non-priority blocks (late predecessors) can delay the acceptance of priority blocks, thereby, as shown in Fig. 4, decreasing the *security threshold*—the minimum fraction of adversarial mining power needed to compromise its liveness—and increasing the transaction confirmation latency even with low bandwidth utilization. A loss of liveness would allow an attacker to arbitrarily confirm or censor transactions. This finding contradicts the authors' claim that the threshold remains constant until the network reaches 90% of capacity.

Our OHIE analysis demonstrates that when the protocol suffers from both block jams and late predecessors, the attacker can undermine not only its liveness, as in our Prism analysis, but also its consistency by reordering confirmed transactions. Compromised consistency would also downgrade the protocol's fairness as the attacker can manipulate both transaction [8] and block order [24] for profit, regardless of how the transaction fees and block rewards are distributed [45]. We validate our analysis with simulations.

II. DAG-based PoW Protocols

Despite being listed among the strongest candidates to break NC's security-performance tradeoff, early DAG-based protocols [5], [20], [21], [36], [37] shy away from the problem as their theoretical analyses are partial and not as rigorous as NC's. Prism [4] and OHIE [44] are the first to explicitly tackle this tradeoff. Their results are encouraging, as they not only analyze their security and performance by extending the mature techniques established for NC, but also, to some extent, decouple security and performance in their analyses. However, these analyses implicitly rely on the Assumption of Decoupling, which may not hold when the underlying protocols operate with high transaction throughput.

A. Proof-of-Work Blockchain Protocols

PoW protocols have been the backbone of most cryptocurrencies [25], and are battle-tested over the past decade, demonstrating their reliability. Alternative protocols, despite their better energy efficiency, all introduce stronger security assumptions, yet none achieves the same level of security [6], [30], [45]. Consequently, a considerable number of recent influential projects are still built on PoW. For example, Grin [16] and Nervos CKB [34] are two chain-based systems launched in 2019; Conflux [31] and Kaspa [33] are two DAG-based systems launched in 2020 and 2021, respectively.

We now describe a PoW blockchain system informally, which suffices to describe our main observation. The formal definitions are delayed to the next section. In a PoW system, each participant maintains its local ledger by executing *the blockchain protocol*; they interact with each other through the P2P network. Miners compete for the rights to extend the ledger by searching for the block candidate whose hash result is below a threshold d. The block candidate consists of (1) some recent transactions, (2) the hash pointers identifying *the predecessors*, i.e., some latest valid blocks, and (3) an arbitrary number called *nonce*, which the miner enumerates to change the hash result. Once a solution is found, the block is broadcast immediately to the network. Upon receiving a block, the miner performs several checks before accepting it and mining on top of it, including (1) whether its hash result is below d, (2) whether all the newly packed transactions are valid, and (3) whether the predecessors are already received and accepted as part of the local ledger. As a block is only valid if all its predecessors are valid, the protocols prescribe miners not to accept blocks with missing predecessors, but to query for them or simply ignore the newly received block.

B. NC and its Security-Performance Tradeoff

NC is the simplest PoW protocol, where each block points to only one predecessor, and the authentic ledger is the *main chain*, i.e., the most computationally challenging chain to produce. The latter rule is commonly, albeit inaccurately, referred to as *the longest-chain rule*. Due to this rule, all but one block choosing the same predecessor are invalid. To ensure security, NC blockchains must reduce these invalid blocks, because they do not contribute to the total computational "work" of the main chain. This total work measures the adversarial mining power threshold to secretly generate a longer chain, hence the system's security. This security requirement translates directly to long block intervals and small block sizes, preventing these systems from reaching the network's physical limits. This tradeoff is analyzed thoroughly in previous works [10], [12]–[15], [19], [29], [35], [38], all of which assume a uniform upper bound of block propagation delay.

Although two chain-based protocols Bitcoin-NG [11] and NC-Max [46] can already process transactions at the network's capacity, their latency still has room for improvement. Bitcoin-NG's latency is identical to that of NC's; Prism outperforms NC-Max in latency when the workload is low [46].

C. Early DAG-based Protocols

To break the tradeoff, DAG-based protocols allow one block to have multiple predecessors, all of which contribute to transaction confirmation. Thanks to these simultaneous blocks, these protocols naturally achieve higher throughput than NC. However, early DAG-based protocols' weak security guarantees or lack of formal analyses renders it difficult to evaluate whether they have broken the tradeoff.

Weak Security Guarantees. The Inclusive protocol [20] adopts the same longest-chain rule with NC, thus inheriting the same tradeoff. Meshcash [5] packs simultaneous blocks into the same layer; these blocks are no longer required to refer to each other. The authors only proved Meshcash's security against an attacker controlling 1/15 of the total mining power.

Partial Security Analyses. SPECTRE [36] forgoes the total order of the blocks and only defines a relative order between any two blocks. Consequently, its security properties are incompatible with those of NC's. PHANTOM [37] improves SPECTRE's design by prescribing a total order. However, its security analysis is against a specific attack strategy, rather than a general attacker. Conflux [21] is a variant of the Inclusive protocol that provides only informal arguments and simulations on its security. Yu et al. [44] demonstrated that the security of Conflux deteriorates quickly as the throughput increases.

D. Prism

Prism [4] and OHIE [44] are the only two exceptions with rigorous proofs related to their security and performance. Their proofs adopt the established proving techniques for NC, thanks to the fact that they also involve chains as their core data structures.[1] We overview their designs here before highlighting how their respective analyses rely on the Assumption of Decoupling.

The Protocol. There are three types of blocks in Prism: transaction blocks, proposer blocks, and voter blocks. A miner mines one transaction block, one proposer block, and m voter blocks—with respective *tree index* numbers from 1 to m—simultaneously by solving the same puzzle. Once a solution

[1] Some studies exclude them from DAG-based protocols, contrary to the authors' claims that Prism is "a structured DAG" (p. 3 in [4]) and OHIE's chains form a DAG (p. 14 in [44]). Agreeing with the authors, we consider them DAG-based because the validity of a block in such a chain often depends on multiple previous blocks, rendering it unprocessable before all of them are received.

is found, a cryptographic sortition function deterministically computes its block type. The block is then released along with the content of its type.

The block content differs based on its type. In a transaction block, the miner packs transactions that have not been packed by other transaction blocks. In a proposer block, the miner packs hash pointers to all transaction blocks and proposer blocks that have not been packed by other proposer blocks. All proposer blocks thus form a *proposer blocktree*, similar to the Inclusive protocol [20]. Each proposer block has a *level*, which is calculated as its longest distance to the earliest proposer block in the proposer blocktree. In a voter block, the miner packs hash pointers to exactly one parent voter block of the same tree index and some proposer blocks. All voter blocks of the same tree index thus form a tree, and each voter block chooses its parent via the longest-chain rule. We say a voter block *votes* for a proposer block if the former includes the hash pointer of the latter. Each voter-block chain votes for exactly one proposer block at each proposer block level.

Prism establishes a canonical order of transaction blocks as follows. As each of the m longest voter chains votes for a proposer block at each level, each proposer block level collects m votes eventually. The proposer block with the most votes on each level is called *leader block*. The leader blocks are then ordered by their level numbers. The sequence of leader blocks for each level is called *leader sequence*. Transaction blocks are ordered by their earliest appearances in the leader sequence. Hence, the leader sequence, which points to transaction blocks using a DAG structure, plays a crucial role in the security of Prism. It is equivalent to the main chain in NC and inherits NC's security properties and analysis.

The Authors' and Later Analyses. Prism's authors claimed that its performance gain comes from "decoupling functionalities". Transaction synchronization is assigned to transaction blocks, which are relatively large and slow, consuming most of the network's capacity. The ordering and confirmation of transaction blocks are assigned to the proposer and voter blocks, respectively, which are small and enjoy priority in propagation. Specifically, in Eqn (9), Sect. 4.2 of [4], the authors computed the propagation delays of proposer and voter blocks, denoted Δ_p and Δ_v, as $\Delta_\mathrm{p} = B_\mathrm{p}/C + D$ and $\Delta_\mathrm{v} = B_\mathrm{v}/C + D$, respectively, where B_p and B_v are the block sizes, C is the network's capacity, and D is the actual (minimum) network delay. Afterward, they further simplified these delays as D since B_p and B_v are small, and then applied this delay throughout their analysis and simulations. As Δ_p and Δ_v are independent of the transaction processing workload, the proposer and voter chains can always grow at the fastest possible rate, enabling Prism to achieve the optimal security threshold of close to 50% and performance close to the physical limits at the same time—the authors used 90% of the network's bandwidth in their examples and simulations.

Li and Guo provided security proofs for Prism under the lockstep [22] and non-lockstep [23] synchronous model without the finite horizon assumption. Yang et al. proposed Prism++, which improves Prism's confirmation rules and the way nodes process parallel messages [43]. These studies also assume independence between the proposer and the voter blocks' acceptance and the transaction blocks' propagation.

We argue that such decoupling does not hold in reality. A proposer/voter block cannot be processed until all the blocks it referred to are received and verified. As the protocol prescribes in Sect. 3.1 [4], "if the miner lacks some referred blocks, it requests them from the network". However, the protocol makes no guarantee that the referred blocks are received before the priority blocks. We will show how this late-predecessor phenomenon influences the leader sequence and hence breaks the independence between Prism's security and performance.

E. OHIE

The Protocol. OHIE only has one type of block, but the protocol maintains m parallel NC instances, all of which follow the longest-chain rule. Similar to Prism, miners in OHIE work on all m chains simultaneously by solving the same puzzle. Once a block is found, the last several bits of its hash determine the chain index.

Compared to NC's blocks, an OHIE block has two additional fields $(\texttt{rank}, \texttt{next\_rank})$, which are used for ordering blocks across chains. The `rank` of a block is the `next_rank` of its same-chain parent block. The `next_rank` is the larger one between (1) $\texttt{rank} + 1$ and (2) the `next_rank` of the block(s) with the largest `rank` the miner is aware of. In the case of (2), an additional hash pointer would be included, which points to the "largest `rank`" block with the smallest chain index, denoted the *trailing block*. Blocks are ordered first by their `rank`, then by their chain index. So OHIE adopts a structured DAG, which runs m parallel NC chains using `rank` for ordering transactions in different chains.

The Authors' Analysis. The authors explained OHIE's performance gain with the following key insight: with a small block size limit—20 KB in the authors' example—and a large number of parallel NC chains, the block propagation delay is short and stable. Such a delay allows the authors to prove that OHIE achieves the optimal security threshold of close to 50% while the throughput increases linearly with m, "until we start to saturate the network bandwidth". The authors verified their claims with simulations.

OHIE's decoupling of security and performance relies on the short and stable block propagation delay, which demands that the network is not saturated. The authors acknowledged this and experimentally showed a "non-trivial increase" in this delay when utilizing more than 50% of the network capacity (Fig. 4 in [44]), confirming the block-jam phenomenon. Nevertheless, their theoretical analysis cannot answer how would this increased delay affect OHIE's security and performance.

III. Congestible Blockchain Model

A. Why a New Model?

UDBM Cannot Model DAG-based Protocols. Most analyses on NC assume a uniform upper bound Δ on the block propagation delay, and the adversary can manipulate the delay arbitrarily within this bound. Consequently, the adversaries, in their optimal strategies, often delay the blocks' propagation to all receivers exactly to this upper bound. We define this *uniform delay blockchain model* as Δ-UDBM:

Definition 1 (UDBM). *Under Δ-UDBM, when a node P_i sends a message to the environment $\mathcal{Z}$ in round r, all nodes will receive the message no later than round $r + \Delta$.*

The rationale behind UDBM is three-fold.

- *The homogeneity of blocks*: NC has only one type of block; all blocks abide by the same block size limit.
- *The independence of block propagation*: when the physical limits are not reached, simultaneous blocks do not affect each other's propagation delay.
- *Overestimating the adversary's ability*: when designing a simplified model in security analyses, we always prefer an adversary stronger than reality to a weaker one.

The last aspect is already discussed in [26], [42] to account for the fact that the adversary has restricted corruption speed and may not always succeed.

We argue that in a DAG-based system, the first two aspects do not hold either. Specifically, a homogeneous-block setting would disregard the protocol designers' efforts in designing multiple types of blocks with different size limits and priority propagation policies, thus underestimating the protocols' security and performance.

Assuming total independence in block propagation is more problematic. As DAG-based systems shorten the block interval, overlaps in block propagation happen frequently and by design, leading to several phenomena that do not align with UDBM:

- *Consecutive blocks mining the same chain*: if the miners of two consecutive blocks are geographically close, or even the same person, both blocks could be valid even if their interval is shorter than D.
- *Block jams*: due to block jams, it is hard to prescribe a universal upper bound on the block propagation delay.
- *Late predecessors*: late predecessors delay a block's acceptance regardless of whether the network is saturated.

We cannot assign a larger Δ in UDBM to cover the worst-case propagation delay because, as we will prove in Lemma 2, such a worse case is not reachable. As we will show in our simulations, such pessimistic modeling would lead to the incorrect conclusion that the performance of DAG is no better than that of NC.

Experimentally Confirming the Block Propagation Dependency. To show the impact of block dependency in their processing delays, we implement Prism with SimBlock [1], a Bitcoin network simulator, and simulate its operation across 200 nodes distributed on five continents, with an average bandwidth of 5 MBps. Each node randomly selects 5 to 11 adjacent nodes; each connection's propagation delay comes from SimBlock's built-in dataset, which is collected from the Bitcoin network. The proposer and transaction blocks are generated at expected intervals of 10 and 5 seconds, respectively. The small proposer blocks point to the latest transaction blocks, which are large and utilize 20% to 100% of the network's capacity in different settings. We record each proposer block's propagation delay and average actual delay, i.e., the interval between the block's generation and the arrival of its latest predecessor at a certain node. The results in Fig. 1 show that the transaction blocks significantly slow down the processing of proposer blocks.

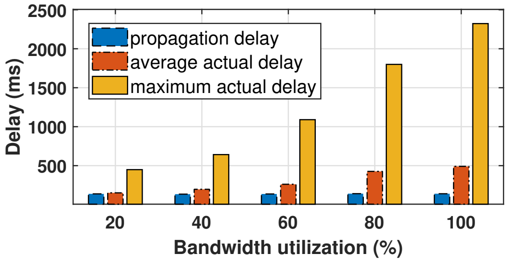

Fig. 1: In Prism, the actual delays of proposer blocks are longer than their own propagation delays, due to the large size of the transaction blocks they refer to.

Other Block Propagation Models. UDBM is generalized via two approaches. The first approach [26], [42] regards each block's propagation as an independent event, overlooking the overlap of these events and the complex block dependencies. The second approach is concurrent with our work and can cover block jams. Proposed by Neu et al. [28] and followed by Kiffer et al. [18], this approach assumes that the block headers are propagated with a known upper bound, and each node can download a fixed number of block contents per time slot. CBM is more general than their model in two aspects. First, our approach—block-specific upper bound—can incorporate the protocols' heterogeneous block types and priority block propagation policies, thus can analyze how these design choices contribute to the protocols' security. Second, CBM has a stronger adversary that can manipulate block propagation within constraints.

B. Our Model

CBM differs from UDBM in two aspects. First, upon each block's generation, the environment outputs the upper bound of its propagation delay in a case-by-case manner. Such flexibility enables the inclusion of many realistic factors including the block size, the network's saturation condition, and the protocols' priority propagation policies. Second, a miner cannot mine on a block until it has received the block's predecessors and earlier priority blocks that jammed the network. This requirement explicitly addresses the complex block-generation events and the block-referring relations in DAG-based protocols.

Timing Assumption. We consider a round-based model commonly used in previous works [4], [19], [29], rather than a continuous-time one as in [15], [35]. This choice does not sacrifice generality, because as argued in [14], [35], these two models are equivalent when the round can be arbitrarily short.

We do not specify how to compute the delay upper bound for each block, but leave it as an external function of the network layer, instantiated by the practitioners. For example, [18], [28] provides one method to set these bounds, abiding by the bandwidth constraint. As a permissionless consensus has no safety guarantee in an asynchronous or partially synchronous environment [29], the upper bound must be chosen from a predefined set of values so that CBM is synchronous.

Mining. There are n *participating nodes* throughout the execution, and each of them has identical mining power. It is possible to model a varying number of participants [13], but we consider it unnecessary as the flat model already reveals many insights. Let κ denote the security parameter. Mining is modeled as querying a random oracle $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, to which all nodes have access. In each round, each node can query $H(x)$ at most once, and verify whether $H(x) = y$ arbitrary times. The input x is a valid block if and only if $H(x) < 2^\kappa \cdot p$, where p is the probability that a query produces a new block. Let f denote the probability that all nodes mine at least one block in a round, then we have $f = 1-(1-p)^n$. When $np \ll 1$, we have $f \approx np$, which is also the expected number of blocks mined by all nodes in a round.

Environment. An *environment* $\mathcal{Z}(1^\kappa)$ captures all aspects in the system external to the protocol Π and the adversary $\mathcal{A}$. $\mathcal{Z}$ keeps the entire mining history and the joint view of all nodes, including their current DAG and all the blocks' processing states, in a variable called the *global state st*.

$\mathcal{Z}$ is also in charge of collecting and sending messages. At the beginning of a round, $\mathcal{Z}$ sends all messages—transactions and blocks—scheduled to deliver to a node in this round to the node, which updates its local state before mining. At the end of a round, all nodes send their newly-mined blocks to $\mathcal{Z}$. The propagation delay is computed collectively by $\mathcal{Z}$ and $\mathcal{A}$, which is detailed next.

Message Propagation Delay. Upon receiving a new block B in a round r, $\mathcal{Z}$ first gives the upper bound of its propagation delay, which is applicable to all nodes, via an abstract function $\delta_{\max}(st, r, B)$. We use $\delta^B_{\max}$ to denote the result. Afterward, $\mathcal{Z}$ sends to the adversary $\mathcal{A}$ a tuple $(st, r, B, \delta^B_{\max})$, who feeds it to a strategy function $\mathcal{S}(\cdot)$ and decides each node's delay $(\delta^B_1, \delta^B_2, \ldots, \delta^B_n)$ for all $i \in [n]$, where $[n]$ denotes $\{1, 2, \ldots, n\}$. $\mathcal{A}$ may also update previous blocks' delivery schedule based on the new mining event. Node i either receives B from $\mathcal{A}$ with delay δ^B_i when $\delta^B_i < \delta^B_{\max}$, or receives it from $\mathcal{Z}$ with delay $\delta^B_{\max}$. This *separation of duty*—$\mathcal{Z}$ decides the upper bound, and $\mathcal{A}$ manipulates arbitrarily within this bound—not only maximizes the adversary's operating space but also enables incorporation of many real-world factors.

Adversary. In any given round, $\mathcal{A}$ controls at most β fraction of the nodes. Nodes not corrupted are *honest*, whose total fraction is α, such that $\alpha + \beta = 1$. $\mathcal{A}$ can corrupt and uncorrupt arbitrarily nodes, but its adaptive corruption has a one-round delay so that he cannot prevent an honest node from sending a newly-mined block to $\mathcal{Z}$. Honest nodes follow the protocol Π; corrupted nodes can mine on anything. $\mathcal{A}$ can delay messages of corrupted nodes arbitrarily, but can only delay and reorder messages from honest nodes up to $\delta_{\max}(\cdot)$.[2] Honest messages cannot be tampered with.

Block Processing. It is necessary to distinguish the following states after a node receives a block: received, accepted, confirmed, and orphaned. *Received* means $\mathcal{Z}$ has delivered the block to the node. *Accepted* indicates that the block has passed the validity check: all the predecessors are accepted, and the block itself is valid. An honest node can only mine after a block after it is accepted. In line with previous works, we omit the time spent in transaction and block validation, as the former is usually executed in parallel with other tasks, and the latter is negligible compared with the network delay. Therefore the only contributor delaying a block's acceptance after receiving is the late predecessors. We use υ^B_i to denote this processing delay, then the actual delay between a block B's generation and its acceptance at node i is $\Delta^B_i = \delta^B_i + \upsilon^B_i$. A block is *confirmed* if it is deemed part of the final ledger; otherwise, the block is *orphaned.*

C. Security Properties

The security of blockchains is defined as *liveness* and *consistency*. In most NC analyses [12], [13], [29], liveness measures how long it takes for a transaction to be confirmed, which is further derived from *chain growth* and *chain quality* properties; consistency measures the difficulty for the adversary to modify the ledger, which is further derived from *common prefix* property. We adopt a similar approach with slightly modified definitions of chain growth.

Let $\mathcal{C}^r_i$ denote chain $\mathcal{C}$ in the view of node i in the beginning of round r, $|\mathcal{C}^r_i|$ denote the length of $\mathcal{C}^r_i$, and $\Delta(\mathcal{C})$ denote the maximum actual delay of blocks on chain $\mathcal{C}$.

Definition 2 (Chain Growth). *We say a chain $\mathcal{C}$ in protocol Π has g-chain growth if there exists some constant c and a negligible function $negl(\cdot)$ such that for every $\kappa \in \mathrm{N}^+$, any $T > c \cdot log(\kappa)$ and $r > \Delta(\mathcal{C})$, the following holds:*

$$\Pr[\min_i |\mathcal{C}^{r+T}_i| - \min_j |\mathcal{C}^r_j| \geq g \cdot T] \geq 1 - negl(T).$$

Unlike previous works [29], [42] where chain growth is the worst-case growth rate $\min_{i,j}\left(|\mathcal{C}^{r+T}_i| - |\mathcal{C}^r_j|\right)$, our chain growth measures the growth rate of the shortest chain(s).

Definition 3 (Chain Quality). *We say a chain $\mathcal{C}$ in protocol Π has μ-chain quality if there exists some constant c and a negligible function $negl(\cdot)$ such that, for every $\kappa \in \mathrm{N}^+$, any $T > c \cdot log(\kappa)$, $r > \Delta(\mathcal{C})$ and $i \in [n]$, the following holds: In any T consecutive blocks of chain $\mathcal{C}^r_i$, there are at least μT honest blocks with a probability of $1 - negl(T)$.*

Definition 4 (Common Prefix). *We say a chain $\mathcal{C}$ in protocol Π has common prefix if there exists some constant c and a negligible function $negl(\cdot)$ such that, for every $\kappa \in \mathrm{N}^+$, any $T > c \cdot log(\kappa)$, $r_2 \geq r_1 > \Delta(\mathcal{C})$ and $i, j \in [n]$, the following holds: for any two honest chain $\mathcal{C}^{r_1}_i$, $\mathcal{C}^{r_2}_j$, it holds that $\mathcal{C}^{r_1}_i$ with the last T blocks removed is a prefix of $\mathcal{C}^{r_2}_j$ with a probability of $1 - negl(T)$.*

IV. Applying CBM to DAG-based Protocols

To demonstrate the value of CBM, we prove some general results applicable to all DAG-based PoW protocols, which also pave the way for our subsequent analysis. We start by formally defining the late predecessors. Afterward, we prove an

[2] It is a common practice to assume that the adversary can slow down block propagation to a predefined bound. In practice, delaying message propagation can be achieved by deploying a considerable number of Sybil nodes in the network, occupying the honest nodes' connection slots but not forwarding any blocks or transactions. This attack is discussed and analyzed in a series of studies, initiated by Babaioff et al. in 2012 [3].

impossibility result that justifies the non-triviality of CBM—it cannot be reduced to UDBM. At last, we show how to maximize the processing delay of a single block, which would inspire the LP attack. In this section, the adversary may tamper with block propagation but has no mining power.

A. Notations

We list our notation naming rules before diving into the proofs. There are two constants: f is the block generation rate, and n is the total number of nodes. The i-th node is denoted as P_i. There are three kinds of delays: δ_i^B is the *propagation delay*, i.e., the interval between a block B's generation and its arrival at P_i; υ_i^B is the *processing delay*, i.e., the interval between B's arrival at P_i and the arrival of its latest predecessors; Δ_i^B is the *actual delay*, which satisfies $\Delta_i^B = \delta_i^B + \upsilon_i^B$. The subscripts of these delays, apart from denoting the receiver node number, can also be "max" to denote their upper bounds applicable to all receivers.

We define two new superscripts to denote blocks with different propagation characteristics: "$*$" indicates potential late predecessor blocks, which usually have large $\delta_{\max}$ and may arrive later than their successors at some nodes; "$+$" indicates potentially affected blocks, which may experience positive processing delays due to their late predecessors. These superscripts can be applied to B, the round number r, and the delays. We number the superscripts such as B^{*1}, B^{*2} to distinguish multiple blocks of the same type.

B. Defining Late Predecessors

Definition 5 (Late Predecessor and Lag Time)**.** *Consider a block B^* which is received by a node P_i at round R^*. We say B^* is* **late** *for node P_i if there exists a block B^+, which is received by P_i at round R^+ such that (1) $B^* \leftarrow B^+$, i.e., B^* is a predecessor of B^+, and (2) $R^* > R^+$. The* **lag time** *between B^* and B^+ at node P_i is defined as $lt(B^*, B^+, P_i) = R^* - R^+$ if B^* is late for node P_i; otherwise, $lt(B^*, B^+, P_i) = 0$.*

A late predecessor occurs if three events happen in the following order: (1) B^+ is generated with B^* as a predecessor; (2) B^+ reaches P_i; (3) B^* reaches P_i. The interval between (2) and (3) is defined as the lag time. There are many possible reasons for this reversed arrival order, including: (1) B^+ is smaller than B^*, (2) B^+ is a priority block but B^* is not, (3) P_i is geographically close to B^+'s miner, and (4) the network is under attack.

The processing delay of B^+ for P_i, denoted υ_i^+, is then the maximum lag time between B^+ and all its predecessors:

$$\upsilon_i^+ = \max_{j \in [l]} \{lt(B^{*j}, B^+, P_i)\}, \tag{1}$$

where l is the number of B^+'s predecessors. Theoretically, it is possible that after receiving all B^+'s predecessors, P_i discovers that it does not have a predecessor to one of them, thus it has to keep waiting before it can process B^+. We omit this "late predecessor's late predecessor" scenario in this study as it happens rarely and complicates our analysis. In other words, P_i starts mining on B^+ after the processing delay.

C. Bounding the Actual Delay

The *maximum actual delay* of B^+, denoted $\Delta_{\max}^+$, is the interval between its generation and acceptance at all the nodes:

$$\Delta_{\max}^+ = \max_{i \in [n]} \{\Delta_i^+\}, \tag{2}$$

where Δ_i^+ is the actual delay of B^+ at P_i. As we already have $\Delta_i^+ = \delta_i^+ + \upsilon_i^+$, where δ_i^+ is the propagation delay of B^+ at P_i, we can combine Eqn. (1), (2), and Definition 5 to get the following lemma:

Lemma 1. *When $\delta_{\max}^+ < \max_{j \in [l]} \{\delta_{\max}^{B^{*j}}\}$, it holds that $\Delta_{\max}^+ \le \max_{j \in [l]} \{\delta_{\max}^{B^{*j}}\}$.*

Intuitively, Lemma 1 states that the maximum actual delay of a block is bounded by the larger value between its own propagation delay and its predecessors' propagation delay.

Analyses adopting UDBM assume that $\Delta_{\max}^+ = \delta_{\max}^+$ so that they do not need to dissect the late-predecessor phenomenon. As our goal is to model this phenomenon, we are interested in the alternative case. When $\Delta_{\max}^+ > \delta_{\max}^+$, the first question we try to answer is *whether $\Delta_{\max}^+$ is globally achievable*, i.e., $\Delta_1^+ = \Delta_2^+ = \cdots = \Delta_n^+ = \Delta_{\max}^+$. This question has a crucial implication for our analysis: if the adversary can cause all the nodes to accept and start mining on B^+ at the same time, we can consider all honest mining power as a unified group, which simplifies our analysis. (Un)fortunately, that is not possible:

Lemma 2. *When $\Delta_{\max}^+ > \delta_{\max}^+$ and P_i mined B^+, the event that for all j that $j \in [n]$ and $j \neq i$, $\Delta_j^+ = \Delta_{\max}^+$ happens with a probability of $1/n$.*

Proof: Suppose that $\Delta_{\max}^+ > \delta_{\max}^+$ and the event in the lemma happens. Then, for any P_j that $j \neq i$, there must be a predecessor block of B^+ that P_j has not received before round $r^+ + \Delta_{\max}^+$, where r^+ is the round when B^+ is mined. So B^+ can only be mined by P_i. According to the mining model, the probability of the event is $1/n$. ■

We gain two sights from Lemma 2.

Insight 1. Since a globally-uniform actual delay is almost impossible for a block with late predecessors, we cannot simply generalize the analyses in UDBM studies to cover the late-predecessor phenomenon. This is because, in these analyses, the worst attack usually delivers all honest blocks to all receivers simultaneously after the maximum delay.

Insight 2. It is non-trivial to maximize the "damage" of a late predecessor. For each potential late predecessor B^*, the attacker must carefully differentiate its arrival times at the nodes, so that some nodes receiving B^* early can act as potential miners of B^+, while others act as B^+'s early receivers and can be affected by B^*'s late arrival. As we shall see in Sect. V-B, there is a tradeoff here: increasing the first group reduces the "victim set"; if the second group grows, B^+ may not be mined or mined late, lowering the damage as well.

D. Propagating One Potential Late Predecessor

Now we study how an adversary can maximize the "damage" of just *one* potential late predecessor B^* in CBM. Specifically, we explore *how* should B^* be propagated, given the proportions of early and late receivers. We leave deciding such proportions and analyzing *all* potential late predecessors to the next section.

The first step is to define and quantify the "damage", i.e., the adversary's utility. To maximize both the number and the lag time of the affected nodes, we measure the adversary's utility corresponding to one potential late predecessor as its average lag time for all nodes. Formally, given two blocks B^* and B^+, between which B^* is mined earlier, the *average lag time* for (B^*, B^+) is

$$\overline{lt}_S(B^*, B^+) = \frac{1}{n}\sum\nolimits_{i=1}^{n} lt(B^*, B^+, P_i), \tag{3}$$

where S denotes the adversary's block propagation strategy.

Note that $\overline{lt}_S(B^*, B^+)$ could be zero even when some nodes receive B^+ first. Recall that the block miners and mining time are chosen by $\mathcal{Z}$ and the propagation strategy S is chosen by $\mathcal{A}$. As $\mathcal{A}$ cannot predict B^+'s miner and mining time r^+, he may miss the opportunity such that under S, B^* does not reach B^+'s miner at r^+, then B^* is not a predecessor of B^+. Consequently, $lt(B^*, B^+, P_i) = 0$ for all $i \in [n]$ by definition. We denote the probability that "B^* has reached B^+'s miner at r^+, thus B^* is a predecessor of B^+" as $\Pr\{B^* \leftarrow B^+\}$.

Our main result regarding $\overline{lt}_S(B^*, B^+)$ is the next lemma, which is essential to developing the attack strategy in Sect. V.

Lemma 3. *Let $R^* = r^* + \delta^*$ and $R^+ = r^+ + \delta^+$ denote the rounds that all nodes have received B^* and B^+, respectively. Assuming that there are $(1-\rho)n$ nodes that have received B^* at round r^+, we have:*

1) $\Pr\{B^* \leftarrow B^+\} = 1 - \rho$ *regardless of how B^* propagates before round r^+;*
2) $\overline{lt}_S(B^*, B^+) \leq \rho \cdot (R^* - R^+)$.

Proof: We define $F^*(r)$ as the fraction of nodes who have received B^* in round r, so we have $F^*(r^+) = 1 - \rho$. The event that $B^* \leftarrow B^+$ only depends on whether the miner of B^+ has received B^* at r^+. Since the probability of winning in a round is equal for all the nodes, we can get $\Pr\{B^* \leftarrow B^+\} = F^*(r^+) = 1 - \rho$, proving (1).

Assuming that $B^* \leftarrow B^+$, we use $F^*(\cdot)$ to calculate $\overline{lt}_S(\cdot)$:

$$\begin{aligned}\overline{lt}_S(B^*, B^+) &= \sum\nolimits_{r=R^++1}^{R^*}[F^*(r) - F^*(r-1)] \cdot (r - R^+)\\ &\leq \rho \cdot (R^* - R^+)\end{aligned}$$

where the inequality follows from $F^*(R^+) \geq F^*(r^+) = 1-\rho$ and $F^*(R^*) = 1$. In fact, there are ρn nodes that have not received B^* at round r^+. If the adversary forces them to all wait until round R^* to receive B^*, their processing delay of B^+ will be $R^* - R^+$. So the maximum is achievable. ■

Intuitively, Lemma 3 states that, given that B^+ is mined at r^+ when $(1-\rho)n$ nodes have received B^*, for any strategy S, there is always a corresponding strategy S' that achieves at least the same, if not better, $\overline{lt}_S$. The variant S' modifies S by sending B^* right after r^* to all nodes that have received it before r^+, and to all other nodes at R^*, hoping that the former modification raises $\Pr\{B^* \leftarrow B^+\}$ and the latter modification extends the lag time for some nodes.

Insight 3. The maximum average lag time concerning one potential late predecessor is achievable by sending it to some nodes as early as possible and to the others as late as possible.

V. Late-Predecessor Attack and the Updated Security Properties

Given that CBM is an abstract model, we introduce an LP attack that outputs the detailed block propagation process for a given protocol and a set of network parameters. Recall that CBM models block jams with block-by-block propagation delay upper bounds determined by the environment rather than the adversary. Therefore, the attack focuses on maximizing the effect of late predecessors. We start by defining the attacker's utility, followed by a description of the attack strategy and its rationale, and quantify the strategy's utility. At last, we show how to update the security properties of a DAG-based protocol in the presence of an LP attacker.

A. Defining the Attacker's Utility

To define a quantifiable utility so that we can gain more insights, in this section, we enforce the same propagation delay upper bound among blocks of the same function/type; blocks with different functions or types can still have different upper-bound delays. Note that our general results regarding CBM in Sect. III do not rely on such homogeneity—all the blocks could have different propagation delays, yet the lemmas still hold. There are two groups of blocks we are interested in: *potential late predecessors*, which may act as late predecessors when they are pointed by an *affected block*—the second group. The set of potential late predecessors is denoted $\mathcal{B}^*$. Although these two groups are mutually exclusive, they may belong to the same type in the protocol's design, as in our OHIE analysis. In line with Sect. IV-A, the superscripts "$*$" and "$+$" correspond to the first and second groups, respectively. When the context is clear, we use B^* and B^+ to denote a member of its group. Let f^* denote the probability that at least one B^* is generated in a round. All blocks in $\mathcal{B}^*$ abide by the same propagation delay upper bound $\delta^*_{\max}$; $\delta^+_{\max}$ is defined likewise, with $\delta^+_{\max} \leq \delta^*_{\max}$.

We explain how we choose the utility with three steps of reasoning. Recall that the average lag time defined in Eqn. (3) considers only one B^*, which may not be the latest. Naturally, our first step is to generalize it to the *average actual delay* of a B^+, which considers *all* its predecessors:

$$\overline{\Delta}^+ = \frac{1}{n}\sum\nolimits_{i=1}^{n}(\delta^+_i + \upsilon^+_i), \tag{4}$$

where the processing delay υ^+_i is the interval between B^+'s and its latest predecessor's arrivals at P_i. Our second step is, $\mathcal{A}$ cannot compute $\overline{\Delta}^+$ directly as it has no knowledge of when or where B^+ will be mined, therefore a strategy output by $\mathcal{A}$ can only maximize the expectation of $\overline{\Delta}^+$. Our third step is to generalize one B^+ to all of them. As a general attack that is agnostic to the specific protocol, there is no knowledge for $\mathcal{A}$ to choose which B^+ should have longer $\mathbb{E}[\overline{\Delta}^+]$. Therefore, we want a uniform $\mathbb{E}[\overline{\Delta}^+]$ throughout the execution. Such equality permits us to choose a homogeneous, i.e., somewhat symmetric, propagation strategy with respect to all blocks in $\mathcal{B}^*$, which also makes $\mathbb{E}[\overline{\Delta}^+]$ computable. Therefore, we choose $\mathbb{E}[\overline{\Delta}^+]$ as $\mathcal{A}$'s utility. Next, we describe such a strategy.

1) When a potential late predecessor B^* is mined in G_j, the adversary executes the following steps:
   - deliver B^* to other nodes in G_j immediately;
   - deliver B^* to the other sets after $\delta^*_{\max}$ rounds.
2) When a B^+ is mined, deliver it after $\delta^+_{\max}$ rounds for all the nodes.

Fig. 2: Late-predecessor attack.

B. The Attack Strategy

Fundamentally, the late predecessor attack generalizes the dominant strategy described in Insight 3 (see Sect. IV-D) to all blocks in $\mathcal{B}^*$, and locates the most effective ρ which balances the tradeoff described in Insight 2 (see Sect. IV-C). We first describe the propagation strategy for one B^*, and then discuss how to generalize it to $\mathcal{B}^*$.

For each B^*, according to Lemma 3, to maximize its average lag time, thus its contribution to $\mathbb{E}[\overline{\Delta^+}]$, $\mathcal{A}$ can select a subset of $(1-\rho)n$ nodes, which we call B^*'s *local set*, and send B^* to them at the beginning of round r^*+1 while setting the propagation delay of other nodes as $\delta^*_{\max}$.

Now we consider a sequence of blocks in $\mathcal{B}^*$. If $\mathcal{A}$ can predict when and where the next B^+ is mined, he would place the miner of the latest block in $\mathcal{B}^*$, which is mined right before B^+, in the same local set of B^+'s miner, to maximize the chance that this "latest B^*" becomes a late predecessor of B^+ for everyone outside this local set. However, as $\mathcal{A}$ has no knowledge of the next B^+, the best he can do is to split the blocks in $\mathcal{B}^*$ into disjoint local sets, hoping that no matter where B^+ is mined, there is a B^* in this local set to act as its late predecessor. Therefore, $\mathcal{A}$ divides all nodes into s disjoint subsets, termed $G_1, G_2, ..., G_s$, where $|G_1| = |G_2| = \cdots = |G_s|$, and each subset is the local set of its nodes, so that for each B^*, $\rho = 1 - 1/s$. The complete strategy is in Fig. 2, with an illustration in Fig. 3.

Practicality and Optimality. In practice, the most challenging step for the adversary is to split the network into a small number of disjoint local sets, to manipulate the inter-set latency. This can be achieved via AS-level attacks, such as [2], [41]. Our attack is easier to pull off than [2], [41] as the local sets here do not need to be fully disconnected from each other.

Our strategy seems suboptimal at first glance, as we allow two consecutive blocks in $\mathcal{B}^*$ to be mined in the same subset. A better strategy is to relocate the second block's miner to another subset, to maximize the chance that one of them "meets" the next B^+. However, as indicated by Matt et al. [26], such right-after-mining node relocation is impractical in practice; network-layer attacks usually take hours, if not weeks, to launch [2], [41]. Therefore, by prescribing stable local sets, the LP attack balances its practicality and optimality.

C. Computing the Attacker's Utility

Now we analyze $\mathcal{A}$'s utility and explore how to choose the parameter s that maximizes it.

Theorem 1. *For any $s \in \mathrm{N}^+$, the LP attack in Fig. 2 can make $1-1/s$ fraction of the nodes accept B^+ after Δ^+ rounds such that $\mathbb{E}[\Delta^+] = \delta^+_{\max} + (k - s(1-\omega))/f^*$, where $\omega = (1 - f^*/s)^{\delta^*_{\max}-\delta^+_{\max}}$ and $k = f^* \cdot (\delta^*_{\max} - \delta^+_{\max})$. The expectation of the average actual delay is*

$$\mathbb{E}[\overline{\Delta^+}] = \delta^+_{\max} + (1 - 1/s)(k - s(1-\omega))f^*.$$

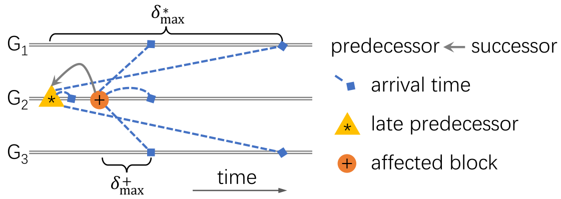

Fig. 3: An example of the attack when $s = 3$. A new B^* reaches its local set G_2 immediately, but to all other nodes as late as possible, hoping that a B^+ will be mined in G_2, whose acceptance will be delayed for nodes in G_1 and G_3.

TABLE I: The optimal s that maximizes $\mathbb{E}[\overline{\Delta^+}]$, where k is the expected number of in-propagation blocks in $\mathcal{B}^*$ in a round.

k	(0.5,2.53)	[2.54,9.81)	[9.82,18.64)	[18.65,20]
s	2	3	4	5

Theorem 1 gives the maximum average delay caused by our attack. In the next two sections, we will prove that the security threshold is negatively related to the average delay. To prove this theorem, we need Lemma 4, 5, and 6.

Lemma 4. *When a B^+ is mined, the expected number of blocks in $\mathcal{B}^*$ that have not finished their propagation is $k = f^* \cdot (\delta^*_{\max} - \delta^+_{\max})$.*

Lemma 5. *When B^+ is mined in G_j for some $j \in [s]$, it holds that*

$$\mathbb{E}[v_i^+] = \begin{cases} 0 & P_i \in G_j \\ \delta^*_{\max} - \delta^+_{\max} - (s/f^*)(1-\omega) & P_i \notin G_j \end{cases},$$

where $\mathbb{E}[v_i^+]$ is the expected processing delay for all possible B^+ and $\omega = (1 - f^*/s)^{\delta^*_{\max}-\delta^+_{\max}}$.

Lemma 6. *Let A_j denote the event that B^+ is mined in G_j, for $j \in [s]$. We have*

$$\Pr[A_1] = \Pr[A_2] = \cdots = \Pr[A_s] = 1/s.$$

The proofs of Lemma 4, 5 and 6 are based on the strategy in Fig. 2 and the properties of the mining process, which follows the Poisson distribution. By the three lemmas, we can calculate the sum of the product of the probability of B^+ being mined by different groups and the corresponding actual delay. Their formal proofs are in Appendix D.

Theorem 1 shows that $\mathbb{E}[\overline{\Delta^+}]$ increases along with $\delta^*_{\max}$ and f^*, i.e., longer propagation delay and more late predecessors lead to longer actual delays of the affected blocks, meeting our intuition. Table 1 displays the optimal s values that maximizes $\mathbb{E}[\overline{\Delta^+}]$. When $0 < k \leq 20$, the optimal s are all near $\sqrt{k}$. When $k > 20$, $(1 - f^*/s)^{k/f^*} < e^{-k/s}$ is negligible. That means, $\mathbb{E}[\overline{\Delta^+}] \approx \delta^+_{\max} + (1-1/s)(k-s)/f^*$.

Due to AM-GM inequality, the maximum is achieved when $s = \sqrt{k}$. Therefore, for any k, the optimal s is $\lfloor \sqrt{k} \rfloor$ or $\lceil \sqrt{k} \rceil$.

D. Security Properties in the Presence of an LP Attacker

Now we incorporate the LP attack in a full security analysis by analyzing how late predecessors downgrade the security properties of a single chain of affected blocks; these results pave the way for our Prism and OHIE analyses. This step is non-trivial as we cannot replace the delay in existing UDBM analyses with the maximum or average actual delay in an LP attack, as nodes in our model have different delays for the same late predecessor.

Chain Growth. When an affected block is mined, if the miner has accepted one of the highest affected blocks, the chain of affected blocks—*affected chain* for short, grows by one; otherwise—sometimes the highest affected blocks are delayed by the late predecessors, and the affected chain does not grow.

Theorem 2 (Chain Growth). *Under the LP attack, for any $\sigma > 0$, the affected chain $\mathcal{C}^+$ has g-chain growth, where $g = (1-\sigma)\gamma f^+$ and $\gamma = \alpha/(1 + \alpha f^+ \mathbb{E}[\overline{\Delta^+}])$.*

Proof (sketch): Consider the block time intervals of the affected blocks. After the generation of each affected block, the honest nodes' computing power is "wasted" until they accept that block. As the honest affected block mining rate is αf^+, by Theorem 1, it is expected that $\alpha f^+ \mathbb{E}[\overline{\Delta^+}]$ blocks are mined during the delay and only one out of $1 + \alpha f^+ \mathbb{E}[\overline{\Delta^+}]$ blocks extends the affected chain. Thus the affected chain growth is at least $(1-\sigma)\alpha f^+/(1+\alpha f^+ \mathbb{E}[\overline{\Delta^+}])$, where σ is the deviation parameter of Chernoff Bound. ■

Incorporating the network propagation process into the proof involves dissecting block mining events into several cases, whose details are in the full proof in Appendix A.

Chain Quality. Chain quality measures the percentage of honest blocks in the affected chain. Intuitively, when $\beta > \gamma$, i.e., when the honest affected chain growth rate is slower than the attacker's mining rate, the attacker can generate a secret chain to invalidate the honest affected chain. Therefore, the affected chain has a positive chain quality only when $\beta < \gamma$:

Theorem 3 (Chain Quality). *If $\beta < \gamma$, the affected chain $\mathcal{C}^+$ has μ-chain quality with $\mu > 0$.*

Proof (sketch): When $\beta < \gamma$, the generation rate of the adversary's affected blocks is slower than the growth rate of the affected leader sequence. Therefore, there exists at least one affected block not mined by the adversary during a long enough period, safeguarding the chain quality. ■

See Appendix B for the full proof.

Common Prefix. We investigate the common prefix property of affected blocks by analyzing the probability of the adversary splitting honest nodes to work on two distinct chains of the same length. An opportunity to end the forked situation arises when (a) an honest affected block extends the longest affected chain. Hopefully, after (a), the honest nodes converge to the same chain after all of them have received this block and all its late predecessors. Therefore, intuitively, it is always rational for the attacker to delay the acceptance of the honest affected block to everyone outside its local set for as long as possible. When the delay ends, only two events can prevent this convergence: (b) an honest node mines a new affected block of the same height on another chain before the delay ends; (c) the adversary possesses a private chain longer than the previous forked chain. The common prefix cannot hold if the union frequency of events (b) and (c) exceeds the frequency of (a). The following lemma formalizes the above intuition and gives the probability that neither (b) nor (c) happens.

Lemma 7. *If $\gamma > \beta$, when the longest affected chain increases by 1, then all honest nodes would converge to the same chain with a probability of at least $(1 - e^{-\Omega(\gamma/\beta)}) \cdot e^{-\Omega(\alpha\delta^* f^+/f^*)}$.*

We get the common prefix by computing the probability that the attacker fails to maintain a fork of length T.

Theorem 4. *For any $T > c \cdot log(\kappa)$, $r_2 \geq r_1 > \Delta(\mathcal{C})$ and $i, j \in [n]$, the affected chain $\mathcal{C}_i^{r_1}$ with the last T blocks removed is a prefix of $\mathcal{C}_j^{r_2}$ with a probability of at least $1 - e^{-T\cdot\Omega(\alpha\delta^* f^+ f^*/\beta)}$, which proves that the affected chain satisfies common prefix.*

Lemma 7 and Theorem 4 are proved in Appendix C.

VI. Prism's Security-Performance Tradeoff

As discussed in Sect. II-D, Prism decouples its security and performance via separation of duty: the proposer and voter blocks ensure security with short and constant propagation delays, while transaction blocks boost performance with large size and parallel processing. We show that increasing the number and size of transaction blocks compromises Prism's security by (1) lowering the growth rate of the proposer blocktree, which further (2) allows an attacker to dominate the proposer leader sequence by invalidating honest blocks. In the extreme case, when the attacker produces proposer blocks faster than the lowered growth rate of the honest proposer blocktree, he can seize the entire proposer leader sequence and censor transactions at will, violating Prism's liveness. Therefore, by neglecting late predecessors, Prism's authors overestimated its *security threshold*, i.e., the minimum adversarial mining power to break its liveness.

The block propagation process of (1) is given by the LP attack, in which the transaction blocks and honest proposer blocks serve as the potential late predecessors and the affected blocks, respectively. We exclude the attacker blocks from the affected blocks, as the attacker can delay them at his discretion; there is no need to distinguish between honest and attacker transaction blocks because they both can be used for the attack. For (2), our analysis of chain quality follows that of Sect. V-D.

A. Applying the Security Analysis

Let δ_{t} and δ_{p} denote the maximum propagation delays of transaction and honest proposer blocks, respectively, where $\delta_{\mathrm{t}} > \delta_{\mathrm{p}}$. The generation rate of transaction and proposer blocks are denoted as f_{t} and f_{p}, respectively. Therefore, the generation rate of honest proposer blocks is αf_{p}. We apply the strategy in Fig. 2 and the security analysis to Prism and instantiate Theorem 1, 2 and 3 to the following corollaries.

Corollary 1. *Under the LP attack, the honest proposer block can be delayed Δ_{p} rounds for $1 - 1/s$ of the honest nodes, and be delayed δ_{p} rounds for the rest $1/s$, where*

$$\mathbb{E}[\Delta_{\mathrm{p}}] = \delta_{\mathrm{t}} - s \cdot (1 - (1 - f_{\mathrm{t}}/s)^{\delta_{\mathrm{t}} - \delta_{\mathrm{p}}})/f_{\mathrm{t}}.$$

Therefore, $\mathbb{E}[\overline{\Delta}_{\mathrm{p}}] = \delta_{\mathrm{p}}/s + (1 - 1/s) \cdot \mathbb{E}[\Delta_{\mathrm{p}}]$.

Corollary 2 (Leader Sequence Growth)**.** *For any $\sigma > 0$, the proposer leader sequence $\mathcal{C}_{\mathrm{p}}$ has g-chain growth, where $g = (1 - \sigma)\gamma f_{\mathrm{p}}$ and $\gamma = \alpha/(1 + \alpha f_{\mathrm{p}} \mathbb{E}[\overline{\Delta}_{\mathrm{p}}])$.*

Corollary 3 (Leader Sequence Quality)**.** *If $\beta < \gamma$, the proposer leader sequence $\mathcal{C}_{\mathrm{p}}$ has μ-chain quality with $\mu > 0$.*

These corollaries update the security threshold of the proposer blocktree, hence the leader sequence, proving that the threshold is not independent of δ_{t}, even if we assume $\delta_{\mathrm{p}} = 0$.

Leader sequence quality determines the liveness of Prism, which is the security property that guarantees any honest transaction must enter the ledger eventually. The adversary controls the full ledger and can launch censorship attacks once the system cannot guarantee $\mu > 0$.

B. Quantifying the Security-Performance Tradeoff

To ensure Prism's liveness, i.e., an honest transaction will eventually be recorded in the ledger, we combine $\alpha + \beta = 1$ and Theorem 3 to get the following threshold for β:

$$\beta < \gamma = \frac{1}{2} - \frac{\sqrt{(f_{\mathrm{p}} \mathbb{E}[\overline{\Delta}_{\mathrm{p}}])^2 + 4} - 2}{2 f_{\mathrm{p}} \mathbb{E}[\overline{\Delta}_{\mathrm{p}}]}. \tag{5}$$

The "Security-CBM" in Figure 4 shows how γ decreases with the increase of $f_{\mathrm{p}} \mathbb{E}[\overline{\Delta}_{\mathrm{p}}]$, which reflects the wasted computing power due to the LP attack. By Corollary 1, $f_{\mathrm{p}} \mathbb{E}[\overline{\Delta}_{\mathrm{p}}]$ is positively correlated with f_{t} and δ_{t}, where the latter depends on the transaction block size $|B_{\mathrm{t}}|$. This counters the authors' claim ("Security-[2]" in Fig.4) that Prism's security is independent of its throughput, computed as $(1 - \beta) f_{\mathrm{t}} \cdot |B_{\mathrm{t}}|$ by the authors, and transaction blocks' propagation delay. Such independence does not hold even when the bandwidth is not exhausted: it is difficult to identify a clear turning point before which γ is stable.

The transaction confirmation latency is not constant, either. This latency is defined by the authors as the time for a transaction to be confirmed with the probability of $1 - \epsilon$, where ϵ is the security parameter. Its expectation is determined by c and Δ_{p}, where c is a constant depending on β. The author used $\Delta_{\mathrm{p}} \approx D$ to prove that the latency only depends on D, which is the "Latency-[2]" in Fig.4. But our result ("Latency-CBM") shows that there is also a tradeoff between throughput and latency, similar to that of security.

Now we compare Prism's original analysis and our results with a set of parameters given by its authors. According to [4], when a block's propagation delay grows linearly with its size, Prism can achieve a throughput of $\lambda = 0.9(1 - \beta)C$, where C is the network's capacity; its transaction ledger achieves consistency and liveness as long as $f_{\mathrm{p}} = f_v < \log((1 - \beta)/\beta)/(1 - \beta)$. Specifically, when $f_{\mathrm{p}} = f_{\mathrm{t}} = 0.1$ block/sec, $B_{\mathrm{t}} = 9C$, $D = 1$ sec, $\delta_{\mathrm{t}} = |B_{\mathrm{t}}|/C + D = 10$ sec, and $\delta_{\mathrm{p}} \approx D = 1$ sec, Prism is safe when $\beta = 0.48$ and the optimal throughput λ is achievable.

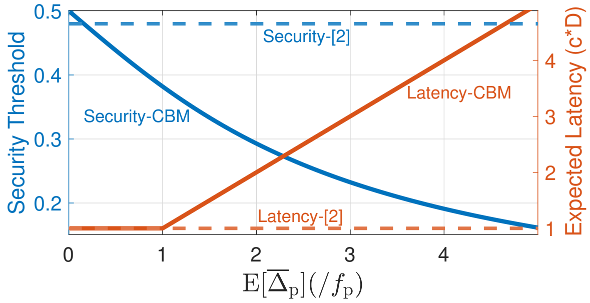

Fig. 4: Fixed confirmation reliability $1 - \epsilon$. Prism's security threshold decreases and transaction confirmation latency increases with increasing damage of the LP attack. $\mathbb{E}[\overline{\Delta}_{\mathrm{p}}]$ is in units of the reciprocal of f_{p} and the unit of expected latency is $c * D$.

Our analysis shows that Prism loses liveness with these parameters. When each round is 0.001 sec and $s = 2$, we have $\mathbb{E}[\overline{\Delta}_{\mathrm{p}}] \approx 2376$ according to Corollary 1, i.e., the proposer blocks' average actual delay is about 2.376 sec, rather than 1 sec claimed by the authors. Consequently, the threshold γ from Eqn. (5) is 0.47, indicating Prism has no resistance against an attacker with $\beta = 0.48$.

VII. OHIE's Security-Performance Tradeoff

We are interested in the case when the block propagation delay is no longer constant, i.e., when the bandwidth utilization exceeds 50% based on the authors' simulations. In this case, OHIE suffers from both block jams and late predecessors. We model block jams with a linear relation and late predecessors with techniques similar to our Prism analysis. The results show that these phenomena aggravate each other's effects and threaten both OHIE's liveness and consistency.

A. Modeling the Two Phenomena

Block Jams. Block jams affect the security-performance tradeoff by extending the block propagation delay, which grows superlinearly according to Fig. 4 in [44] when the bandwidth utilization exceeds 50%. Therefore, we use a linear function, which is in favor of honest miners, to demonstrate how OHIE's security deteriorates under high performance.

As the bandwidth utilization grows with the number of parallel chains m, we model the block propagation delay upper bound δ as

$$\delta = \mathrm{c}_{\mathrm{bj}} \cdot \max\{m - \mathrm{m}_{\mathrm{nc}}, 0\} + D \ , \tag{6}$$

where D is the network propagation delay, c_{bj} is a constant coefficient related to **b**lock **j**ams, and m_{nc} is a threshold, representing the maximum number of chains that will **n**ot **c**ause a block jam.

Late Predecessors. To apply the attack to OHIE, the first task is to choose the potential late predecessors. We cannot choose a type of block directly as in our Prism analysis, because OHIE has only one block type. Ideally, we want a small number of late predecessors to affect as many blocks as possible. Therefore, we make our choice based on the following observation. Between the two predecessors of a typical OHIE block, the parent block is often referred to only once, while the trailing block, i.e., the largest `rank` block with the smallest chain index the miner is aware of, is often referred to by multiple blocks. As a result, delaying a *potential trailing block*, i.e., the earliest block with a new `rank`, has a larger chance to affect multiple blocks. Consequently, we choose these blocks as the potential late predecessors. Formally, we use B_r to denote a block that is mined earlier than all other blocks of its `rank`, and $\mathcal{B}_\mathrm{r}$ to denote the whole set of B_rs. As in our Prism analysis, we assume the propagation delay of all other blocks to be zero.

Next, we define the affected blocks. Let f_r denote the generation rate of $\mathcal{B}_\mathrm{r}$ for the whole system, which increases along with m. Recall that the adversary partitions all nodes into s equal-sized sets $G_1, \cdots, G_s$, thus the generation rate of $\mathcal{B}_\mathrm{r}$ in each set is f_r/s. A B_r mined in G_j will be referred to by all subsequent honest blocks mined in G_j, at least before the next block in $\mathcal{B}_\mathrm{r}$ is mined. Therefore, we define these honest blocks as B_r's affected blocks. We use B_s to denote one of these **s**uccessors.

B. Applying the Attacks and the Security Analysis

We denote the mining rate for *each chain* as f. We only consider the case when the bandwidth utilization exceeds 50%, so Eqn. (6) becomes $\delta = \mathrm{c_{bj}} \cdot (m - \mathrm{m_{nc}}) + D$. We give the following lemma to calculate f_r and leave its mathematical derivation in Appendix E.

Lemma 8. *For $f > 0$ and $m > 0$,*

$$f_\mathrm{r} = \frac{1}{\int_0^{+\infty} m f^2 x^2 (1+fx)^{m-1} e^{-mfx} dx}$$

which increases as m does.

Due to the LP attack, any new block cannot be accepted by most nodes until the end of the delay for their trailing blocks. By Theorem 1, we have

Corollary 4. *The honest blocks can be delayed Δ rounds for $1 - 1/s$ fraction of all honest nodes, with*

$$\mathbb{E}[\Delta] = \delta - s/f_\mathrm{r} + s(1 - f_\mathrm{r}/s)^{\delta}/f_\mathrm{r}.$$

Therefore, $\mathbb{E}[\overline{\Delta}] = (1 - 1/s)\mathbb{E}[\Delta]$.

Departing from Theorem 2, we update OHIE's chain growth:

Corollary 5. *Under the LP attack, the chain growth rate for each chain in OHIE is $\alpha f/(1 + \alpha f \mathbb{E}[\overline{\Delta}])$.*

Unlike Prism, OHIE requires the honest nodes to defeat the attacker in the chain growth rate of *every* chain, to guarantee the chain quality and common prefix properties. This is because the attacker can change the total order of transactions as long as he invalidates some confirmed blocks in any one of the m chains. In other words, as long as the discounted honest chain growth rate $\alpha/(1+\alpha f\mathbb{E}[\overline{\Delta}])$ becomes smaller than β, both OHIE's liveness and its consistency are broken. Let T be the number of blocks that OHIE removes from the end of the chain to obtain the confirmed blocks. We apply Theorem 3 and 4 to each chain of OHIE and calculate the union probability that every parallel chain satisfies security properties. Following the authors' notations (Theorem 1 of [44]), we have the following corollary.

Corollary 6. *Under the LP attack, if $\beta < \alpha/(1 + \alpha f \mathbb{E}[\overline{\Delta}])$, OHIE protocol satisfies Consistency and Quality-Growth with probability at least $1 - m \cdot e^{-\Omega(\kappa)} - m \cdot e^{-\Omega(T)}$.*

C. Tradeoff

According to the authors, OHIE achieves better performance by increasing the value of m. However, our analysis reveals that increasing m decreases OHIE's security threshold via three mechanisms. First, it raises the block propagation delay δ (Eqn. (6)), which increases $\mathbb{E}[\overline{\Delta}]$ (Corollary 4), thus lowering the threshold of each chain (Theorem 5). Second, by increasing the number of chains, the total number of affected blocks grows, thereby increasing the likelihood that the attacker controls a chain. Third, it increases f_r, leaving a higher number of attack targets (Lemma 8). This is counterintuitive as increasing m does not change the growth rate of any single chain. Here is the reason behind this counterintuitive phenomenon. At any given moment, consider the block with the highest `rank`, which is mined in G_j. The highest `next_rank` advances as long as one node in its *receiver set* mines a block, be it G_j or all the nodes. This receiver set mines m chains simultaneously, and for each of these m chains, the expected interval follows an exponential distribution. Therefore, the minimum interval of these m exponential distributions decreases with an increasing m. Due to space constraints, we omit the theoretical analysis but instead present the results along with the simulations.

VIII. Simulations

A. Setup

We modify SimBlock by adding 1000 LoC to evaluate Prism's and OHIE's concrete security-performance tradeoff. For comparison, we also simulate NC and include our theoretical analyses under both CBM and UDBM.

We setup a P2P network of $n = 200$ nodes, each connecting to $d = 8$ randomly selected peers. This network thus forms a random graph, allowing us to estimate the average number of hops between two nodes as $\log n / \log d \approx 2.5$. We cap the per-node bandwidth by $C = 8$ Mbps and set the latency between any two connected nodes as 0.4 sec. Therefore the average delay between two random nodes is $D = 0.4 \times 2.5 = 1$ sec, in line with Prism's setup.

Prism. According to the authors, "choosing large B_t and small f_t is preferable" [4]. Therefore, our simulation has mining rate $f_\mathrm{p} = f_\mathrm{t} = 0.1$ block/sec. We set $|B_\mathrm{t}| = 2, 4, 6, 8$ and 10 MB, to get 20% to 100% bandwidth utilization. The propagation delays of the transaction and the proposer blocks are $\delta_\mathrm{t} = |B_\mathrm{t}| \cdot 2.5/C + D$ and $\delta_\mathrm{p} \approx 1$ sec, respectively.

OHIE. We choose the same set of parameters as in [44]. The block interval on each chain is 10 sec, and the block size is 20 KB. We set the number of parallel chains m from 1 to 500.

When $m < 250$, the propagation delay is about 1 sec. Otherwise, due to block jam, we apply Eqn. (6) with $\mathrm{m_{nc}} = 250$ (corresponding to 50% bandwidth) and $\mathrm{c_{bj}} = 0.08$ to get the propagation delay. The latter is learned via maximum likelihood estimation from OHIE's simulations [44]. The bandwidth utilization of OHIE is thus $m \times \text{block\_size}/\text{block\_interval}/C$.

NC. The block interval is 10 sec, and the block size $|B|$ is 2 to 10 MB, corresponding to 20% to 100% bandwidth utilization. The propagation delay for NC is also $\delta = |B|/C \cdot 2.5 + D$.

Theoretical Results. We also calculated some theoretical security thresholds with the above sets of parameters. First, we calculated these thresholds based on our analyses to test their reasonableness. Specifically, the theoretical versions of Prism and OHIE are denoted "Prism-CBM" and "OHIE-CBM", whose results are calculated from Theorem 2 and 5, respectively. Second, we computed their UDBM variants, denoted "Prism-UDBM" and "OHIE-UDBM", by simply replacing the actual delays in these theorems with their upper bounds, i.e., δ_{t} and Eqn. (6) for Prism and OHIE, respectively. At last, we calculated the threshold based on Prism's original analysis, which is a constant 0.487 regardless of the bandwidth utilization; the corresponding curve is denoted "Prism- [4]".

B. Results

The security-performance tradeoff is visualized as the relationship between bandwidth utilization and security thresholds. We display the results of Prism and OHIE in Fig. 5 and Fig. 6, respectively.

We gain three insights from Prism's results. First, our theoretical analysis meets well with our simulation, demonstrating the model's reasonableness. Second, Prism is not exempt from the security-performance tradeoff. Even if the proposer block is small enough so that its average propagation delay is only 1 sec, the security threshold is not constant as the authors hoped. The security threshold is 0.39 and 0.38 when the bandwidth utilization reaches 90% and 100%, respectively. The 90% result is different from our estimation in Sect. VI-B merely due to a difference in the inputs. Specifically, in Sect. VI-B, we use $\delta_{\mathrm{t}} = |B_{\mathrm{t}}|/C + D$ as in the authors' analysis, because they assume the network is a complete graph, and thus every node is reachable within one hop. In contrast, we assume a random graph, thus $\delta_{\mathrm{t}} = |B_{\mathrm{t}}|/C \cdot 2.5 + D$ in Prism-CBM. Third, UDBM downgrades Prism's security to that of NC's. In UDBM, the proposer blocks propagate as slowly as transaction blocks, underestimating the protocol's security and performance. The gap between the models demonstrates that our model is more accurate and reasonable than UDBM in evaluating the DAG-based protocols' security and performance.

Our OHIE results reveal two additional insights. First, we confirm the authors' claim that smaller and more frequent blocks indeed decouple OHIE's security and performance. When the throughput is below 50% bandwidth, late predecessors do not downgrade the security threshold, which remains stable at 0.47. Second, once the throughput exceeds 50% bandwidth, the block jams undermine the protocol's security to a level similar to that of Prism. Specifically, the threshold is about 0.37 at 100% bandwidth.

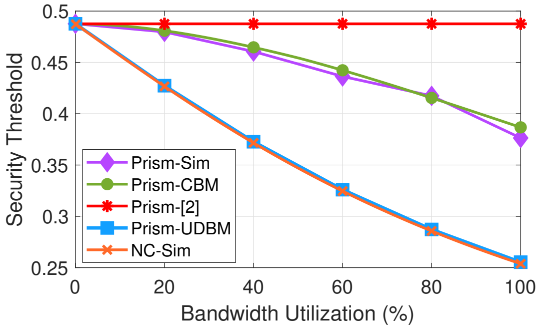

Fig. 5: Prism: Security threshold vs. bandwidth utilization. Due to the late predecessors, Prism's security is not independent of its performance as the authors hoped. Also, UDBM significantly underestimates Prism's security.

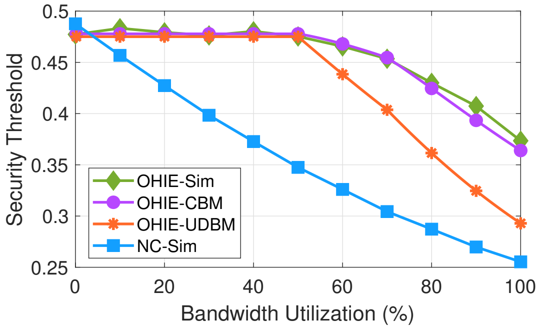

Fig. 6: OHIE: Security threshold vs. bandwidth utilization.

IX. Discussion

Generalizability of the Tradeoff. The blocks' propagation overlaps with each other in all DAG-based protocols, including PoS protocols, such as LMD-GHOST of Ethereum 2.0 [7], and BFT protocols, such as DAG-Rider [17], Tusk [9], and Bullshark [39]. Consequently, as long as the Assumption of Decoupling does not hold, which usually happens in high-throughput settings, the blocks' processing delay is extended, downgrading the system's security properties. CBM and the LP attack can thus be generalized to quantify this tradeoff in these protocols. Specifically, PoS and partially synchronous BFT protocols may be analyzed in a similar manner, except that the block intervals no longer follow exponential distributions. Asynchronous protocols present more challenges as their normal-case delays are usually not analyzed.

Generalizability of CBM. Beyond analyzing DAG-based PoW protocols by modeling the complex block-referring relations and propagation overlap, CBM covers two other aspects of blockchain designs outside the scope of previous blockchain models. First, honest nodes in the network are not treated as a whole but receive the same message with different delays. Therefore, our analysis in Sect. IV and V, when applied to NC and its variants, provide additional insights than the parallel work [18]. Second, CBM supports multiple types of blocks with different propagation policies, which enables the formal analyses of Bitcoin-NG [11], Strongchain [40], and Flux [32].

Practicality and Optimality of Our Attack. Our goal is to reveal the security-performance tradeoff that was overlooked in DAG-based protocols in a reasonable network model. Specifically, the LP attack is considered a plausible worst-case scenario of the late-predecessor phenomenon. Our simulation could be made closer to reality by replacing the delay set by the attacker with actual attacks that delay the blocks' propagation.

The attack strategy favors simplicity and compatibility over optimality. Hence, we do not assert that it is the optimal attack against Prism and OHIE. We leave protocol-specific optimal attacks to future work.

Improving DAG-based Protocols. Our results do not preclude the possibility of future DAG-based protocols that overcome this tradeoff. We encourage protocol designers to pay attention to the network layer and to explicitly address the two aforementioned phenomena, either by introducing extra mechanisms to avoid them or by removing the reliance on the Assumption of Decoupling. For example, we can enforce an interval between transaction synchronization and confirmation through careful engineering as in NC-Max [46], a chain-based protocol. Prior to that, practitioners should only execute the protocols within a safe range of parameters, rather than pushing the network's capacity and sacrificing security.

X. Conclusion

DAG-based PoW protocols are among the most promising candidates for reaching the network's physical limits. However, their security guarantees are unclear, as most existing protocols lack rigorous analysis. Here we presented CBM, which captures the key factors affecting block propagation and acceptance in DAG-based protocols. We further proposed an LP attack in CBM that exploits these factors to delay the blocks' acceptance in DAG-based protocols. We applied this attack to Prism and OHIE, two representative protocols that claim to break the security-performance tradeoff, and update their security proofs and simulations under CBM. Our results show that these protocols are not immune to the tradeoff: their security and performance degrade when operating at high throughput.

References

[1] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops*, 2019, pp. 325–329.

[2] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy, SP 2017*. IEEE Computer Society, 2017, pp. 375–392.

[3] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On Bitcoin and red balloons," in *13th ACM Conference on Electronic Commerce*. ACM, 2012, pp. 56–73.

[4] V. K. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*. ACM, 2019, pp. 585–602.

[5] I. Bentov, P. Hubácek, T. Moran, and A. Nadler, "Tortoise and Hares consensus: the Meshcash framework for incentive-compatible, scalable cryptocurrencies," *IACR Cryptology ePrint Archive*, 2017. [Online]. Available: https://eprint.iacr.org/2017/300.pdf

[6] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proceedings of the 2019 ACM Conference on Economics and Computation*, ser. EC '19, 2019, pp. 459–473.

[7] V. Buterin, D. Hernandez, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining GHOST and Casper," *CoRR*, vol. abs/2003.03052, 2020. [Online]. Available: https://arxiv.org/abs/2003.03052

[8] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.

[9] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus," in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys '22, 2022, p. 34–50.

[10] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Everything is a race and Nakamoto always wins," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2020, pp. 859–878.

[11] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 45–59. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

[12] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, ser. Lecture Notes in Computer Science, vol. 9057. Springer, 2015, pp. 281–310.

[13] ——, "The bitcoin backbone protocol with chains of variable difficulty," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science, vol. 10401. Springer, 2017, pp. 291–323.

[14] P. Gaži, A. Kiayias, and A. Russell, "Tight consistency bounds for bitcoin," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. ACM, 2020, p. 819–838.

[15] D. Guo and L. Ren, "Bitcoin's latency–security analysis made simple," in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 22*, 2022, p. 244–253.

[16] T. E. Jedusor and A. Poelstra, "Grin," 2019. [Online]. Available: https://grin.mw/

[17] I. Keidar, E. Kokoris-Kogias, O. Naor, and A. Spiegelman, "All you need is DAG," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021, p. 165–175.

[18] L. Kiffer, J. Neu, S. Sridhar, A. Zohar, and D. Tse, "Security of nakamoto consensus under congestion," Cryptology ePrint Archive, Paper 2023/381, 2023. [Online]. Available: https://eprint.iacr.org/2023/381

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

[20] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security - 19th International Conference, FC 2015*, ser. Lecture Notes in Computer Science, vol. 8975, pp. 528–547.

[21] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*, pp. 515–528.

[22] J. Li and D. Guo, "On analysis of the Bitcoin and Prism backbone protocols in synchronous networks," in *57th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2019, pp. 17–24.

[23] ——, "Liveness and consistency of Bitcoin and Prism blockchains: The non-lockstep synchronous case," in *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020*. IEEE, 2020, pp. 1–9.

[24] D. Malkhi and P. Szalachowski, "Maximal extractable value (MEV) protection on a DAG," 2022. [Online]. Available: https://arxiv.org/abs/2208.00940

[25] mapofcoins, "Map of coins: BTC map," 2018. [Online]. Available: http://mapofcoins.com/bitcoin

[26] C. Matt, J. B. Nielsen, and S. E. Thomsen, "Formalizing delayed adaptive corruptions and the security of flooding networks," in *Advances in Cryptology – CRYPTO 2022*, 2022, pp. 400–430.

[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[28] J. Neu, S. Sridhar, L. Yang, D. Tse, and M. Alizadeh, "Longest chain consensus under bandwidth constraint," Cryptology ePrint Archive, Paper 2021/1545, 2021. [Online]. Available: https://eprint.iacr.org/2021/1545

[29] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, ser. Lecture Notes in Computer Science, vol. 10211, 2017, pp. 643–673.

[30] A. Poelstra, "Distributed consensus from proof of stake is impossible," 2014. [Online]. Available: https://dergigi.com/assets/files/Poelstra2014.pdf

[31] Conflux Project, "Conflux network," 2020, https://confluxnetwork.org/.

[32] Flux Development Team, "Flux: the decentralized computational network," 2022, https://whitepaper.app.runonflux.io/.

[33] Kaspa Project, "Kaspa," 2021, https://kaspa.org/.

[34] Nervos Foundation, "Nervos network," 2019. [Online]. Available: https://www.nervos.org/

[35] L. Ren, "Analysis of nakamoto consensus," Cryptology ePrint Archive, Paper 2019/943, 2019. [Online]. Available: https://eprint.iacr.org/2019/943

[36] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," Cryptology ePrint Archive, Report 2016/1159, 2016. [Online]. Available: https://ia.cr/2016/1159

[37] Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM GHOSTDAG: a scalable generalization of Nakamoto consensus: September 2, 2021," in *AFT '21: 3rd ACM Conference on Advances in Financial Technologies*. ACM, 2021, pp. 57–70.

[38] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in Bitcoin," in *Financial Cryptography and Data Security - 19th International Conference, FC 2015*, ser. Lecture Notes in Computer Science, vol. 8975. Springer, 2015, pp. 507–527.

[39] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: DAG BFT protocols made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22, 2022, p. 2705–2718.

[40] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun, "Strongchain: Transparent and collaborative proof-of-work consensus," in *Proceedings of the USENIX Security Symposium*, 2019, pp. 819–836.

[41] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang, "A stealthier partitioning attack against Bitcoin peer-to-peer network," in *2020 IEEE Symposium on Security and Privacy, SP 2020*. IEEE, 2020, pp. 894–909.

[42] P. Wei, Q. Yuan, and Y. Zheng, "Security of the blockchain against long delay attack," in *Advances in Cryptology - ASIACRYPT 2018*, ser. Lecture Notes in Computer Science, vol. 11274. Springer, 2018, pp. 250–275.

[43] L. Yang, X. Wang, V. Bagaria, G. Wang, M. Alizadeh, D. Tse, G. Fanti, and P. Viswanath, "Practical low latency proof of work consensus," 2023. [Online]. Available: https://arxiv.org/abs/1909.11261

[44] H. Yu, I. Nikolic, R. Hou, and P. Saxena, "OHIE: blockchain scaling made simple," in *2020 IEEE Symposium on Security and Privacy, SP 2020*. IEEE, 2020, pp. 90–105.

[45] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *40th IEEE Symposium on Security and Privacy (S&P)*. IEEE, May 2019, pp. 1190–1207.

[46] R. Zhang, D. Zhang, Q. Wang, S. Wu, J. Xie, and B. Preneel, "NC-Max: Breaking the security-performance tradeoff in Nakamoto consensus," *The Network and Distributed System Security (NDSS) Symposium*, 2022.

Appendix A
Proof of Theorem 2

Proof: Assume that there exists an honest affected block B_0 mined at round r_0, and its actual delay is Δ_0. Let $d_0 = height(B_0)$. We have $d_0 \geq \min_{i\in[n]} |\mathcal{C}_i^{r_0}|$. Suppose that B_1, which is mined at round r_1, is the first honest affected block satisfying $height(B_1) = d_0 + 1$. Let $\theta_1 = r_1 - r_0$. Next, we show that $\mathbb{E}[\theta_1] \leq 1/\alpha f^+ + (1 - 1/s)\Delta_0$. Consider the following three cases for B_0, where θ_1 in case a is denoted as $\theta_{1,a}$ for $a \in \{1, 2, 3\}$.

- Case 1: B_0 is the first and only honest affected block with height d_0. In this case, once accepting B_0, all honest nodes will mine after B_0. But $(1 - \frac{1}{s})n$ honest nodes will accept B_0 after $r_0 + \Delta_0$ round under the LP attack. Denote the set of these delayed honest nodes as G_{B_0}, and the rest of honest nodes as G'_{B_0}. There are two cases for B_1: (a) B_1 is mined by G'_{B_0} between round $r_0 + 1$ and round $r_0 + \Delta_0$; (b) B_1 is mined after round $r_0 + \Delta_0$ by G_{B_0} or G'_{B_0}. Denote the probability that G'_{B_0} mines an affected block in a round as p'. We have $p' = \alpha f^+/s$ and

  $$\begin{aligned}&\Pr[\theta_{1,1} = x]\\ &= \begin{cases} p'(1-p')^{x-1}, & x \leq \Delta_0 \\ (1-p')^{\Delta_0}(1-p's)^{x-\Delta_0-1}p's, & x > \Delta_0 \end{cases}\end{aligned}$$

  So we get

  $$\begin{aligned}&E[\theta_{1,1}]\\ &= \sum_{x=1}^{\Delta_0} x \cdot \Pr[\theta_{1,1} = x] + \sum_{x=\Delta_0+1}^{+\infty} x \cdot \Pr[\theta_{1,1} = x]\\ &= \frac{1}{p'} + (1-p')^{\Delta_0}(\frac{1}{p's} - \frac{1}{p'})\\ &\leq \frac{1}{p'} \cdot [1 + (1 - \Delta_0 p')(\frac{1}{s} - 1)]\\ &\leq \frac{1}{\alpha f^+} + (1 - \frac{1}{s})\Delta_0,\end{aligned} \tag{7}$$

  where the second equality follows from the geometric series formula and the expectation of geometric distribution, and the first inequality follows from Bernoulli inequality.

- Case 2: B_0 is the first honest affected block with height d_0 but there will be other competing honest affected blocks, say $B_{01}, B_{02}, \ldots, B_{0j}$. Let G_0^r denote the set of all honest nodes that accept at least one of honest affected blocks with height d_0 at round r, i.e., $G_0^r = G_{B_0} \cup (\cup_{i=1}^{j} G_{B_{0i}})$, where $j + 1$ is the number of existing honest affected blocks with height d_0 in round r. We have $|G_0^r| \geq |G_{B_0}| = n/s$ for any $r > r_0$. Similar to the analysis of case 1, we get $\mathbb{E}[\theta_{1,2}] \leq \mathbb{E}[\theta_{1,1}] \leq 1/(\alpha f^+) + (1 - 1/s)\Delta_0$.

- Case 3: If B_0 is not the first honest affected block with height d_0, there exists an honest affected block B'_0, which is the first honest affected block with height d_0. Suppose that B'_0 is mined at round r'_0, where $r'_0 \leq r_0$. According to the above analysis, we get $\mathbb{E}[r_1 - r'_0] \leq$

$1/(\alpha f^+)+(1-1/s)\Delta_0$. So we have $\mathbb{E}[\theta_{1,3}] = \mathbb{E}[r_1 - r_0] \leq \mathbb{E}[r_1 - r_0'] \leq 1/(\alpha f^+) + (1-1/s)\Delta_0$.

To sum up, we have $\theta_1 \leq 1/(\alpha f^+)+(1-1/s)\Delta_0$. Similarly, let B_i be the first honest affected block satisfying $height(B_i) = d_0+i$ for $i = 1, 2, \dots, t$. Suppose that r_i is the round when B_i is mined, and their actual delay is Δ_i. Let $\theta_i = r_i - r_{i-1}$ and $\theta^* = \sum_{i=1}^{t} \theta_i$. Using the Chernoff bound, we have $\Pr[\theta^* < (1+\sigma')\cdot\mathbb{E}[\theta^*]] > 1-e^{-\Omega(\sigma'^2 t)}$. Since $\mathbb{E}[\theta^*] \leq t\cdot(1/(\alpha f^+)+(1-1/s)\mathbb{E}[\Delta^+]) = t\cdot(1/(\alpha f^+)+\mathbb{E}[\overline{\Delta}^+])$, we have

$$\Pr[\theta^* < (1+\sigma')\cdot t\cdot(1/(\alpha f^+)+\mathbb{E}[\overline{\Delta}^+])] > 1-e^{-\Omega(\sigma'^2 t)} \quad (8)$$

which shows that the length of affected leader sequence increases by at least t between round r_0 and round r_t. Notice that all honest nodes can accept B_0 at round $r_0+\Delta(\mathcal{C}^+)$, and all honest nodes can accept B_t at round $r_t+\Delta(\mathcal{C}^+)$. Hence, we have $\min_i |\mathcal{C}_i^{r_t+\Delta(\mathcal{C}^+)}| - \min_j |\mathcal{C}_j^{r_0+\Delta(\mathcal{C}^+)}| \geq t$.

Due to (8) and $r_t - r_0 = \theta^* = T$, we can get the following inequality by choosing appropriate σ.

$$\Pr[\frac{(1-\sigma)\cdot T}{1/\alpha f^+ + \mathbb{E}[\overline{\Delta}^+]} < \min_i |\mathcal{C}_i^{r_t+\Delta(\mathcal{C}^+)}| - \min_j |\mathcal{C}_j^{r_0+\Delta(\mathcal{C}^+)}|] \\ > 1 - negl(T),$$

where $negl(T) = e^{-\Omega(\sigma^2 gT)} \geq e^{-\Omega(\sigma^2 t)}$. So we have

$$g = \frac{(1-\sigma)\alpha f^+}{1+\alpha f^+\mathbb{E}[\overline{\Delta}^+]} = (1-\sigma)\gamma f^+$$

with the probability of $1 - negl(T)$, where $\gamma = \frac{\alpha}{1+\alpha f^+\mathbb{E}[\overline{\Delta}^+]}$. ■

Appendix B
Proof of Theorem 3

Proof: Suppose that the length of affected chain increases by t from round r to round $r+T$. By Chernoff bound and Theorem 2, we have

$$Pr[T < (1+\sigma_1)\frac{t}{\gamma f^+}] > 1 - e^{-\Omega(\sigma_1^2 T)}$$

for $\sigma_1 > 0$. Suppose that the adversary mines t_{a} blocks during the T rounds. Note that t_{a} follows Bernoulli distribution $\mathbf{B}(T, \beta f^+)$ with $E[t_{\text{a}}] = \beta f^+ T$. According to Chernoff bound, we have

$$Pr[t_{\text{a}} < (1+\sigma_2)\beta f^+ T] > 1 - e^{-\Omega(\sigma_2^2 T)}$$

for $\sigma_2 > 0$. Combining the two inequalities, we have

$$Pr[t_{\text{a}} < (1+\sigma_2)(1+\sigma_1)\frac{\beta t}{\gamma}] > 1 - e^{-\Omega(\sigma_1^2 T)} - e^{-\Omega(\sigma_2^2 T)}.$$

When $\beta < \gamma$, we can choose appropriate small σ_1, σ_2 such that

$$(1+\sigma_1)(1+\sigma_2)\frac{\beta}{\gamma} < (1+\sigma)\frac{\beta}{\gamma} < 1.$$

Then we have

$$Pr[1 - t_{\text{a}}/t > 1 - (1+\sigma)\frac{\beta}{\gamma} > 0] > 1 - negl(T).$$

Since the adversary cannot modify the contents of messages sent by honest nodes, there are at least $[1-(1+\sigma)\frac{\beta}{\gamma}]t > 0$ blocks mined by honest miners, which completes the proof. ■

Appendix C
Proof of Lemma 7 and Theorem 4

Define the chain whose last block is mined by an honest node as an honest chain, and the length of the longest honest chain as *global height*. If a chain's last blocks are all mined by the adversary, the end of the chain that is entirely composed of adversary blocks is defined as an adversary fork. The existence of such forks is unknown to honest nodes and they can be published at any time by the adversary. For any honest chain, a chain with the same length but different blocks in the last T blocks is referred to as its T-balanced chain. Such a T-balanced chain may either be an honest chain or a published adversary fork.

Proof of Lemma 7. When an honest node belonging to group G_j mines a new affected block B^+ at round r and the block increases global height from H to $H+1$, we define the inverse events of events (b) and (c): $(\bar{b})$ no honest nodes that do not belong to G_j have mine an affected block in the following Δ^+ rounds; $(\bar{c})$ any adversary fork's length is smaller than H in round $r+\Delta^+$.

It is obvious that if the event $(\bar{b})$ and $(\bar{c})$ happen simultaneously, there cannot be a balanced chain of length $H+1$ in round $r+\Delta^+$. So all honest nodes would converge to the longest chain of length $H+1$. By the mining model and Theorem 1, the probability of events $(\bar{b})$ is

$$\Pr[(\bar{b})] = (1-(1-1/s)\alpha f^+)^{\Delta^+} \approx e^{-(1-1/s)\alpha f^+(\delta^* - s/f^*))}$$

By theorem 3, we know that within t rounds, if the global length has grown by T, then the adversary has mined less than T blocks with a probability of $1-e^{-\Omega(T\gamma/\beta)} > 1-e^{-\Omega(\gamma/\beta)}$. This means that the adversary cannot mine an adversary fork by itself to maintain the balance. So the adversary may start mining its adversary fork on a longer chain and wait for another shorter honest chain to catch up with the length, which can give the adversary a starting advantage. However, the probability of obtaining this advantage is not high since the adversary cannot predict which chain the next honest block will appear on. It happens if and only if the adversary mines on a higher chain and the other chain can catch up within Δ^+ rounds. The probability is at most $1-(1-(1-1/s)\alpha f^+)^{\Delta^+}$. Finally, the probability that the adversary doesn't mine a block from round r to round $r+\Delta^+$ is $(1-\beta f^+)^{\Delta^+}$ So we have that the probability of the event $(\bar{c})$ is

$$\Pr[(\bar{c})] \geq (1-e^{-\Omega(\gamma/\beta)})\cdot(1-(1-1/s)\alpha f^+)^{\Delta^+}\cdot(1-\beta f^+)^{\Delta^+} \\ \approx (1-e^{-\Omega(\gamma/\beta)})\cdot e^{-(1-\alpha/s)f^+(\delta^* - s/f^*)}$$

Since the event $(\bar{b})$ and $(\bar{c})$ are independent, let $s = \sqrt{(1+\alpha)2\alpha\delta^* f^*/2}$, we have that all honest nodes would converge to the same chain with a probability of at least $(1-e^{-\Omega(\gamma/\beta)})\cdot e^{-\Omega(\alpha\delta^* f^+ f^*)}$.

Proof of Theorem 4 Denote the height of the chain in an honest node P_i's view at round r_1 as $d = height(\mathcal{C}_i^{r_1})$. We assume that $d > k$ since the theorem holds obviously if $h \leq k$.

If all honest nodes converged to the same chain between height $d-k$ and height d in round $r_0 < r_1$, then we have that $\mathcal{C}_i^{r_0} = \mathcal{C}_j^{r_0}$. By Lemma 7, there is no convergence during the process of global length increasing by at least T with a probability of at most $[1-(1-e^{-\Omega(\gamma/\beta)}) \cdot e^{-\Omega(\alpha\delta^* f^+ f^*)}]^T$. If the adversary cannot mine more blocks than the honest chain growth at round r_2, all honest will have the same prefix $\mathcal{C}_i^{r_0}$ in the future. This event happens with a probability of $e^{-\Omega(T\gamma/\beta)}$ by Theorem 3. So the probability that $\mathcal{C}_i^{r_1}$ with the last T blocks removed is a prefix of $\mathcal{C}_j^{r_2}$ is approximately $1-e^{-T\cdot\Omega(\alpha\delta^* f^+ f^*/\beta)}$. The theorem holds.

APPENDIX D
PROOF OF LEMMA 4, 5 AND 6

Proof of Lemma 4. If a block B^* is mined before round $r^+ - (\delta^*_{\max} - \delta^+_{\max})$, we have $lt(B^*, B^+, P_i) = 0$ for any $P_i \in \{1, ..., n\}$, which means B^* cannot be late. So only blocks generated between round $r^+ - (\delta^*_{\max} - \delta^+_{\max})$ and r^+ can become late predecessors. Hence, the expected number of these blocks is $f^* \cdot (\delta^*_{\max} - \delta^+_{\max})$, where f^* is the mining rate of potential blocks.

Proof of Lemma 5. Notice that the nodes in G_j have received the same blocks. When B^+ is mined by some node of G_j, there are no fresh blocks with B^+ for nodes in G_j. So $\upsilon_i^+ = 0$ for $P_i \in G_j$.

The processing delay of B^+ for other sets is determined by the round when G_j mines its last potential block $B^{*j,last}$, which is denoted as $r^{*j,last}$. The other sets except G_j will receive $B^{*j,last}$ at round $r^{*j,last} + \delta^*_{\max}$, and receive B^+ at round $r^+ + \delta^+_{\max}$. So these sets will accept B^+ at round $\max\{r^{*j,last} + \delta^*_{\max}, r^+ + \delta^+_{\max}\}$. Let $X_j = r^+ - r^{*j,last}$ and $\delta_{\text{gap}} = \delta^*_{\max} - \delta^+_{\max}$. By definition 5, we have

$$\upsilon_i^+ = ft(B^{*j,last}, B^+, i) = \begin{cases} \delta_{\text{gap}} - X_j & \delta_{\text{gap}} > X_j \\ 0 & \delta_{\text{gap}} \leq X_j. \end{cases}$$

Since δ_{gap} is a constant, the probability of $\delta_{\text{gap}} - X_j$ is determined by X_j. Then we have

$$\mathbb{E}[\upsilon_i^+] = \sum_{x=1}^{\delta_{\text{gap}}} \Pr[X_j = x] \cdot (\delta_{\text{gap}} - x). \tag{9}$$

In our attack, we say G_j gets a potential block if the potential block is a) mined by G_j or b) mined by G_0 and the adversary randomly selects j. Let f_j^* denote the probability that G_j gets at least one potential block in a round. So we have

$$f_j^* = f^* \cdot (\frac{1-z}{s} + z \cdot \frac{1}{s}) = \frac{f^*}{s}.$$

Note that $X_j = x$ implies that G_j gets a potential block in round $r^+ - x$, and has no block from round $r^+ - x + 1$ to $r^+ - 1$. Since X_j follows the geometric distribution with parameter f_j^*, we have $\Pr[X_j = x] = f_j^*(1-f_j^*)^{x-1}$. Due to (9), we have

$$\begin{aligned}
\mathbb{E}[\upsilon_i^+] &= \sum_{x=1}^{\delta_{\text{gap}}} \Pr[X_j = x](\delta_{\text{gap}} - x) \\
&= \sum_{x=1}^{+\infty} \Pr[X_j = x](\delta_{\text{gap}} - x) - \sum_{x=\delta_{\text{gap}}+1}^{+\infty} \Pr[X_j = x](\delta_{\text{gap}} - x) \\
&= \delta_{\text{gap}} - \sum_{x=1}^{+\infty} f_j^*(1-f_j^*)^{x-1}x + \sum_{d=1}^{+\infty} f_j^*(1-f_j^*)^{\delta_{\text{gap}}+d-1}d \\
&= \delta_{\text{gap}} - \frac{1}{f_j^*} + (1-f_j^*)^{\delta_{\text{gap}}} \sum_{d=1}^{+\infty} f_j^*(1-f_j^*)^{d-1}d \\
&= \delta_{\text{gap}} - \frac{1}{f_j^*} + (1-f_j^*)^{\delta_{\text{gap}}} \frac{1}{f_j^*} \\
&= \delta^*_{\max} - \delta^+_{\max} - (1-\omega)\frac{s}{f^*},
\end{aligned}$$

where $\omega = (1-\frac{f^*}{s})^{\delta_{\text{gap}}}$ and $d = x - \delta_{\text{gap}}$. The fifth equality holds since d follows the geometric distribution.

Proof of Lemma 6. Since there are n/s nodes in any G_j, every G_j has $1/s$ of total computational power. According to the mining model, we have that B^+ is mined by G_j with the probability $1/s$ for $j \in \{1, \ldots, s\}$.

Proof of Theorem 1. Without loss of generality, we assume B^+ is mined in G_j, i.e., A_j happens. When A_j happens, the number of nodes that are not in G_j is $n - |G_j| = n(1-1/s)$. By Lemma 5, we can get the first part of Theorem 1. According to (4), Lemma 4 and Lemma 6, we have

$$\begin{aligned}
\mathbb{E}[\overline{\Delta}^+] = & \textstyle\sum_{m=1}^{s} \Pr[A_m] \cdot [(n/s) \cdot \delta^+_{\max} \\
& + n(1-s) \cdot (\delta^+_{\max} + \mathbb{E}[\upsilon_i^+])]/n \\
= & \delta^+_{\max} + (1-1/s)[\delta^*_{\max} - \delta^+_{\max} - s(1-\omega)/f^*] \\
= & \delta^+_{\max} + (1-1/s)[k - (1-\omega)s]/f^*
\end{aligned} \tag{10}$$

which completes the proof.

APPENDIX E
PROOF OF LEMMA 8

Proof: The mining procedure of each chain in OHIE is the same as that of the Nakamoto consensus. So the block interval in each chain follows the exponential distribution with parameter f. Consider the scenario that, after a new trailing block appears, two new blocks are generated in one of the m chains. According to the definition of the trailing block, the first new block will catch up with the new trailing block, and the second block will become the next trailing block. So the shortest time to mine two consecutive blocks in each of the m chains is the generation interval of B_{r}. On the other hand, the time to mine two consecutive blocks follows the sum of two exponentially distributed random variables with parameter f. That is, it follows a Gamma distribution with shape parameter 2 and rate parameter f. By Minimal Order Statistics, the probability density function of the generation interval t_{r} of B_{r} is

$$p(t_{\text{r}}) = m \cdot [1 - F(x, f, 2)]^{m-1} p(x, f, 2) = mf^2x(1+fx)^{m-1}e^{-mfx},$$

where $F(x, f, 2)$ and $p(x, f, 2)$ are the cumulative distribution function and probability density function of Gamma distribution with shape parameter 2 and rate parameter f, respectively. So we can get the mathematical expectation of the generation interval of B_{r} as below.

$$\mathbb{E}[t_{\mathrm{r}}] = \int_0^{+\infty} m f^2 x^x (1 + fx)^{m-1} e^{-mfx} dx.$$

For any $m \geq 1$, we have

$$\frac{d\mathbb{E}[t_{\mathrm{r}}]}{dm} = \int_0^{+\infty} f^2 x^2 [(1+fx)^{m-1} e^{-fx} (1+m(\ln(1+fx)-fx))] dx < 0, \tag{11}$$

where the inequality follows $\ln(1 + x) < x$ for any $x > 0$. That means, $\mathbb{E}[t_{\mathrm{r}}]$ decreases as m increases. Therefore, we get $f_{\mathrm{r}} = 1/\mathbb{E}[t_{\mathrm{r}}]$ increases as m does. ■