








# Phantasm: Adaptive Scalable Mining Toward Stable BlockDAG

Zijian Zhang , *Member, IEEE*, Xuyang Liu , *Member, IEEE*, Kaiyu Feng , *Member, IEEE*, Mingchao Wan , *Member, IEEE*, Meng Li , *Senior Member, IEEE*, Jin Dong , *Member, IEEE*, and Liehuang Zhu , *Senior Member, IEEE*

**Abstract**—Blockchain technology builds an immutable and append-only ledger in peer-to-peer networks, which attracts attention from various fields. However, traditional chain-based blockchain systems typically have the problem of low throughput, leading to unsatisfactory performance. Among the proposed solutions, introducing a structure of the Directed Acyclic Graph (DAG) into the blockchain reaches a high transaction throughput. Such an approach enables blocks to refer to more than one previous block, thus processing blocks in parallel with better performance. However, existing DAG-based blockchain schemes do not establish a deterministic rule for block reference priority. Adversaries can initiate a splitting attack to select block references to affect DAG topology, making the consensus unstable. In this article, we propose a more stable consensus protocol named Phantasm, aiming to stabilize the ordering result in the consensus protocol. The referred blocks can be decided after computing a solution to the block puzzle and the difficulty of this solution affects the number of block references. We design two strategies to guide the honest nodes to select references so that they can resist the splitting attacks to stabilize the ordering. Theoretical analysis and simulation experiments show that Phantasm is more stable than the classic DAG-based blockchain consensus protocol Phantom regarding the ordering results.

**Index Terms**—Block reference strategy, DAG, splitting attack, consensus, blockchain.

Manuscript received 23 May 2023; revised 3 September 2023; accepted 2 October 2023. Date of publication 16 October 2023; date of current version 12 June 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62172040, in part by Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing under Grant GJJ-23-006, in part by the National Key Research and Development Program of China under Grants 2021YFB2701202 and 2022YFB2702402, in part by the National Natural Science Foundation of China (NSFC) under Grants 62372149 and 62002094, and in part by Anhui Provincial Natural Science Foundation under Grant 2008085MF196. Recommended for acceptance by H. Karatza. (Corresponding authors: Jin Dong; Meng Li.)

Zijian Zhang, Xuyang Liu, Kaiyu Feng, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100811, China (e-mail: zhangzijian@bit.edu.cn; liuxuyang@bit.edu.cn; fengkaiyu@bit.edu.cn; liehuangz@bit.edu.cn).

Mingchao Wan and Jin Dong are with the Beijing Academy of Blockchain and Edge Computing, Beijing 100093, China (e-mail: wanmingchao@chainmaker.org; dongjin@baec.org.cn).

Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology, Ministry of Education, Hefei 230009, China, also with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, also with the Anhui Province Key Laboratory of Industry Safety and Emergency Technology, Hefei 230009, China, and also with the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230009, China (e-mail: mengli@hfut.edu.cn).

Digital Object Identifier 10.1109/TSC.2023.3322203

## I. INTRODUCTION

**L**IMITATIONS of traditional blockchain systems on high latency and low throughput hinder their adoption and application. For instances, Bitcoin [1] handles only 7 transactions per second on average, while Ethereum [2] processes 20 transactions per second. An effective solution is to apply Directed Acyclic Graph (DAG) [3], [4], [5] to blockchain systems to avoid the consumption of competitive transactions caused by linear sequenced blocks. In DAG-based blockchains, the traditional chain structure is extended to graph structure, where each block can refer to multiple successor blocks. Therefore, every user can freely choose prior blocks for expansion instead of competing to mine the next block. At the same time, systems can process multiple blocks in parallel, which highly increase the throughput. Due to the superior performance of DAG-based blockchain systems, they have been widely adopted in finance [6], healthcare [7], Internet of Vehicles [8], Internet of Things [9], etc.

However, bringing in DAG structure also increases the risk of computing power dispersion in blockchain systems. Adversaries are more likely to concentrate their computation power on constructing an alternative block. This form of attack is often referred to as 51% attacks in chain-based blockchain systems [10]. Such attacks have been proven to be a potential threat [11], [12] as they can cause forks in blockchain structure and finally lead to a series of issues such as double spending problem [13]. In DAG-based blockchain systems, these attacks typically target the reversal of block order in the graph structure, which result in serious consequences. We exemplify it in Fig. 1. Assuming that the two accounts Alice and Bob in the network each have an initial balance of 100. Alice initiated two transactions with Bob in total during the process of generating blocks  $a, b, c, \dots, m, n, o$ . One of the transactions is for Alice to transfer half of the balance to Bob, which is packaged in block  $b$ . Another transaction is for Alice to transfer 40 to Bob, which is packaged in block  $c$ . If  $b$  is prior to  $c$ , Alice's final balance is 10, and Bob's final balance is 190. If  $c$  is prior to  $b$ , then Alice's final balance is 30 and Bob's final balance is 170. From the graph, we can see that the order of blocks  $b$  and  $c$  directly affects the order of transactions, which leads to differences in both Alice's and Bob's final account balance. This means that if the adversary can reverse the order of  $b$  and  $c$  through attacks, she can achieve changes in account balance throughout the entire networks, which is disastrous in blockchain systems.

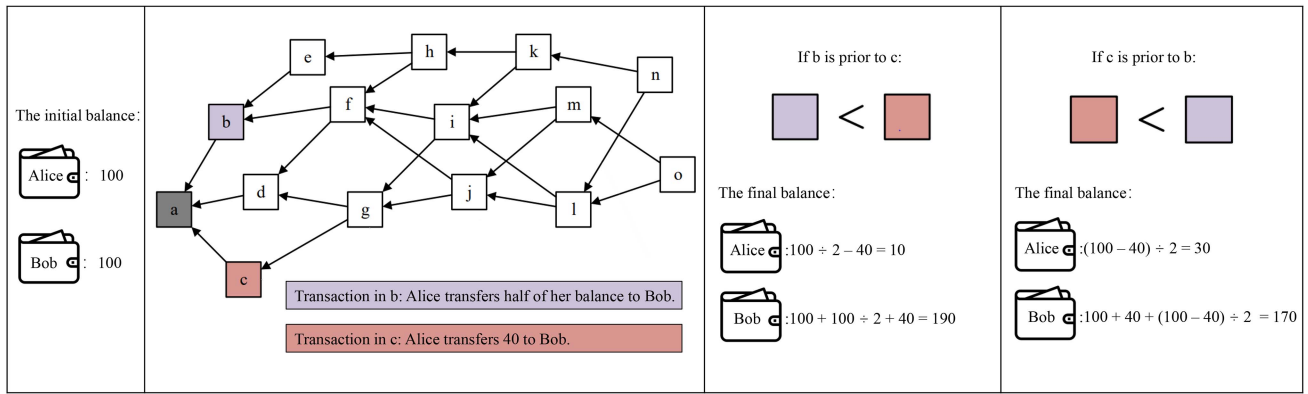


Fig. 1. The impact of block order reversal in DAG-based blockchain systems.

Commonly, the attacks caused by computational power dispersion are often referred to as splitting attacks [14], [15], [16]. In chain-based blockchain systems, since the previous block is unique, the order of blocks is deterministic and clear. Miners can concentrate their computing power on the same point under the longest chain rule to keep the blockchain unique and stable. Therefore, the launch of splitting attacks in chain-based blockchains requires adversaries to possess the majority of the computing power (51% attacks) of the entire networks [17]. But now that the DAG-based blockchains allow each block to refer multiple previous blocks in parallel, the order of those parallel blocks is uncertain and unstable. Miners have the option to select different blocks to continue mining, distributing their computing power over multiple blocks. If the adversaries attempt to alter the ordering of one of these blocks, they only need to possess more computational power than the total power of the miners currently working on that specific block. Due to the dispersion of miners' computing power, the adversary's required computational power is evidently less than half of that of the entire network to successfully launch a splitting attack. Taking Fig. 1 as an example, consider the process of generating 14 blocks  $b, c, d, \dots, m, n, o$ , and the adversary's attack target is block  $b$ . The number of blocks directly or indirectly referencing  $b$  is 10, so the proportion of the computing power required by the adversary to create an alternative block for launching a splitting attack is  $11/(14 + 11) = 44\%$ . This is still based on the premise that there are no references to the alternative block by honest miners. In fact, if the alternative block is referenced by honest miners, the actual computational power required will be smaller. We provide a formal analysis of this example in Section IV.

To alleviate the problem of splitting attacks, state-of-art schemes propose various ordering protocols to resolve the order uncertainty of those unconnected blocks, thus enhancing stability. For examples, Spectre [18] treats blocks as votes to decide the order of any pairwise blocks, where the block with more voting support is sorted first. Phantom [19] conducts voting operations to blocks and invokes a recursive greedy algorithm to sort all the blocks from the genesis block to the latest blocks. Since all blocks refer to the genesis block, the genesis block is the heaviest one. The heavier a block is, the prior the block is ordered. This heaviest chain rule is used to order the sequence of

blocks. It is similar as using the longest chain rule to handle block forks.

Unfortunately, existing solutions still cannot effectively resist splitting attacks. As most of the schemes force the mined blocks to randomly reference as many blocks as possible to increase the probability of being confirmed, this still leads to the dispersion of honest nodes computing power. Adversaries can concentrate their computation power on constructing an alternative block, and conduct the splitting attacks to make consensus results unstable based on the heaviest chain rule.

**Contributions:** In this paper, we propose a new consensus protocol named Phantasm based on the existing DAG-based consensus protocol named Phantom [19]. The Phantasm presents an adaptive scalable mining protocol that supports using the block reference strategies to adjust the DAG topology in a positive way. Under the same ordering protocol of the Phantom, Phantasm can make the consensus results more stable and resist splitting attacks. The concrete contributions of this paper are summarized as follows.

- We explore a new splitting attack against DAG-based blockchain systems: Adversaries concentrate their computation power on constructing an alternative blocks using selfish mining strategy to reverse the block order in Phantom.
- We propose a new DAG-based blockchain consensus protocol named Phantasm. It reconstructs the algorithm for selecting previous blocks by implementing an adaptive scalable mining method that allows the use of block reference strategies to affect the DAG topology. We design two concrete block reference strategies for Phantasm, which assigns priority to each block.
- We theoretically analyze the safety of Phantasm. The simulation verification shows that the Phantasm under both two block reference strategies indeed makes consensus more stable against splitting attacks than the Phantom protocol.

The rest of the paper is organized as follows. We show the related works in Section II and the preliminaries in Section III. The models, observations and principles are established in Section IV. Section V presents the proposed Phantasm protocol, including the adaptive scalable mining and two block reference strategies. We make formal proofs of Phantasm security in

Section VI and take the simulation verification in Section VII. Finally, conclusion is drawn in Section VIII.

## II. RELATED WORK

Different from the blockchain in the chain-based fashion [20], [21], DAG-based blockchains concurrently process multiple blocks, reaching a very high throughput. The consensus schemes generally include a mining protocol and an ordering protocol, where the former builds a BlockDAG and the latter sorts the blocks. According to the topology of the BlockDAG, DAG-based blockchains [5] can be classified into three categories: convergence, parallel, and divergence. We present the mainstream schemes as follows.

The convergence consensus in DAG-based blockchains accepts blocks in multiple forks but gradually converges them into the main chain. GHOST [22], [23], named Greedy Heaviest Observed Sub-Tree, treats the forks as the body of a tree-based chain. It introduces the weight to evaluate the importance of the sub-tree and thus uses the heaviest sub-tree instead of the longest chain [24] to decide the main chain. Inclusive [25] proposes a variant of GHOST, allowing each block to include several uncle blocks except for the unique parent block. It designs an incentive mechanism to encourage miners to include more uncle blocks. Byteball [26] forms a main chain in the graph topology via trustful and reputable witness nodes. A total of twelve witness nodes issue the blocks to decide the main chain index. Conflux [27] presents an adaptive weight for blocks to resist the liveness attack. Each block has one parent edge and multiple reference edges to build the graph, where the heaviest sub-graph can decide the pivot chain to order the blocks. StreamNet [28] requires a new attached block to include two ancestor blocks as references. One of the ancestors is deemed as the parent block, which is determined by the highest score child of sub-tree to build a pivot chain. The other ancestor is randomly selected through the Markov Chain Monte Carlo method to scale out.

The parallel consensus is generally organized by miners with multiple parallel chains, of which the chain is maintained by one miner. Hashgraph [29], [30] creates several separate chains for each miner who issues an event to record the current history and cross-refer other's gossip information. By a three-stage procedure (namely see, strongly see, and decide), this scheme can reach a conventional Byzantine fault-tolerant consensus. Nano [31] uses an account record all the transaction history in a separate chain, where a complete transaction consists of a send transaction and a receive transaction to make a cross-reference. Blockmania [32] executes a practical Byzantine fault-tolerant protocol in parallel chains. Each node only proposes one block on the given position rather than a sequence number, and the blocks jointly form the DAG network. Prism [33] decouples the functionalities of Nakamoto consensus to multiple chains. Blocks are divided into a transaction block, voter block, and proposer block, where each type consists of a separate chain. OHIE [34] allocates each block into an individual chain according to the identity computed by block hash, and the block also refers to another chain for specifying the rank and making the order. JHdag [35] builds a peer chain, the main chain, and

connection chains in the parallel mode. Each miner specifies the block issued by itself to form the peer chain, still includes a milestone block to construct the main chain, and lastly selects another miner's common block to enhance connectivity.

The divergence consensus allows blocks to arbitrarily refer to others, which are structured as a natural expanding graph. Spectre [18] uses a recursive weighted-voting algorithm to decide the pairwise ordering of two blocks. The algorithm completes the voting procedure via blocks instead of miners. Phantom [19] utilizes a recursive k-cluster algorithm to identify the well-connected blocks via coloring the blocks in blue, where the other blocks are excluded and treated as malicious blocks. A greedy algorithm is also presented to compute a full topological order in the blue set of blocks. Iota [6], [36] adopts the UTXO model and establishes the DAG-based blockchain through transactions issued by devices in the Internet of Things. Each transaction is required to approve two ancestor transactions, which are selected by the random walk approaches. GraphChain [37] has a similar design to Iota in which a transaction must verify at least two ancestor transactions. It introduces an incentive mechanism where each transaction contains a fee and chooses a group of transactions with more rewards to deplete their deposit. Meshcash [38] builds the DAG in a layer fashion, where each block points to the blocks in the previous layer. If the current layer sees more than threshold blocks, the counter of the layer increase one. The system utilizes the off-chain asynchronous byzantine agreement protocol to ensure final consistency.

The types of DAG-based blockchain consensus mentioned above are not limited by the traditional chain-based structure anymore. They include as many blocks as possible to increase the system capacity. We are highly concerned about the divergence consensus because such type of scheme has better performance in transaction throughput. To overcome the instability of ordering results, we try to present a new mining protocol, which contains block reference strategies, to effectively adjust the DAG topology. The DAG will grow in the direction followed by the majority of honest users, and thus make the consensus more stable.

## III. PRELIMINARIES

In this section, we briefly introduce a DAG-based blockchain system named Phantom [19], showing the DAG of blocks, mining protocol, and ordering protocol. The DAG of blocks defines the data structure of blocks, the mining protocol present how users create blocks to expand the DAG, while the ordering protocol inputs the DAG and outputs an ordered list of blocks.

### A. The DAG of Blocks

The DAG of blocks (also named BlockDAG) is denoted as  $G = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  represents a set of blocks and  $\mathcal{E}$  represents a set of reference relations each of which consists of two blocks. There is only the unidirectional reference in the BlockDAG, i.e., if block  $B$  refers to block  $A$  (denoted by  $A \leftarrow B$ ), then no reference exists from block  $A$  to block  $B$  (denoted by  $B \leftarrow A$ ). Each block can contain more than one reference and the number of its being referred is also not limited. Besides, we define that



block  $A$  is reachable from block  $B$  if and only if there has a reference path from block  $B$  to block  $A$ . Obviously, the genesis block is reachable from all blocks in the DAG. Here we give four definitions as follows.

- $past(B, G)$  denotes the subset of blocks that is reachable from block  $B$ .
- $future(B, G)$  denotes the subset of blocks from which block  $B$  is reachable.
- $anticone(B, G)$  denotes the subset of blocks outside  $past(B, G)$ ,  $future(B, G)$ , and itself.
- $tips(G)$  denotes the subset of blocks that is not referred to yet.

### B. The Mining Protocol

The block structure contains the header part and body part, where only the header part is used to compute a solution of the block puzzle. Each miner is required to solve such a puzzle for generating a valid block. For simplification, we let the block header mainly include data fields of previous\_hashes, tx\_root, difficulty, and nonce. The hash of block  $B$  is computed below.

$$hash(B) = hash(B.previous\_hashes || B.tx\_root || B.difficulty || B.nonce). \quad (1)$$

The block body contains data fields of transactions, which builds a Merkle tree with the root matching to tx\_root.

The only difference from general block structure is that the data field of previous\_hashes is an array storing multiple block references. A miner takes the current BlockDAG as input to fill data fields for a block (leave empty for nonce), where this block refers to all tips (i.e., leaf blocks) in the current DAG. Then the miner repeatedly attempts random nonce and computes the hash value of the block. With the predefined target hash value  $T$  and the target block difficulty  $D$ , the block  $B$  is valid if

$$T/hash(B) > D. \quad (2)$$

Once found a solution (i.e., nonce) to the puzzle, the miner broadcasts the block and updates the current BlockDAG.

### C. The Ordering Protocol

This protocol commonly has two stages. First, it divides the blocks in the DAG into Blues and Reds (named block coloring). The Blue set represents blocks issued by the majority of honest computing power, while the Red set represents blocks created by the minority of malicious miners. Second, it determines the order according to the topological sort, breaking ties by the block with the lowest hash (named block sorting). For consistency, only blocks in the Blue set are used to make the final ledger. The honest nodes always immediately publish the mined block and refer to other received blocks.

The ordering protocol introduces a parameter  $k$  of cluster factor to estimate the average number of created blocks in each round. The set of blocks created by honest nodes ought to be

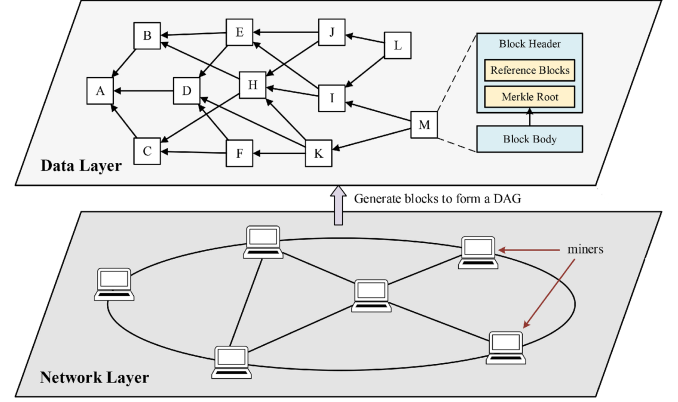


Fig. 2. System model.

well-connected, in which each block's anticone is typically below  $k$ . Thus, the Blue set should satisfy the maximum  $k$ -cluster subgraph.

**Definition 1. (Maximum  $k$ -cluster subgraph):** Given a BlockDAG  $G = (\mathcal{V}, \mathcal{E})$ , a subset  $S \subseteq \mathcal{V}$  is called a  $k$ -cluster if for any  $B \in S$  has  $|anticone(B, G) \cap S| \leq k$ .

Here, the subset  $\mathcal{V} - S$  is the Red set. The ordering protocol uses the above definition to find the maximum  $k$ -cluster subgraph to output a Blue set and adopts a greedy algorithm to sort the subgraph from the genesis block to the current tips.

## IV. PROBLEM STATEMENT

In this section, we give an overview of the DAG-based blockchains, which at a high level describes the system and threat model, our observations and principles to inspire the protocol design.

### A. System Model

A DAG-based blockchain system usually contains two layers, which are the network layer and the data layer. The network layer consists of miners connected to each other, and they generate and broadcast blocks to construct the data layer. We show the system model in Fig. 2. In the network layer, each miner randomly connects several miners as the neighbor so that it builds a peer-to-peer network. The miners adopt the gossip protocol to spread new blocks and transactions.

The data layer is abstracted from the blocks stored by miners, where each block has multiple references to previous blocks. The miners depict the direct acyclic graph of blocks by the reference relations. Due to the network latency, different miners may have different views of this DAG. They will finally get the global view of that after syncing with others. Fig. 2 illustrates an example of the data layer, and we list the DAG of blocks according to the definition in Section III.

$$past(H, G) = \{A, B, C\}.$$

$$future(H, G) = \{I, J, K, L, M\}.$$

$$anticone(H, G) = G - (past(H, G) + future(H, G) - H)$$

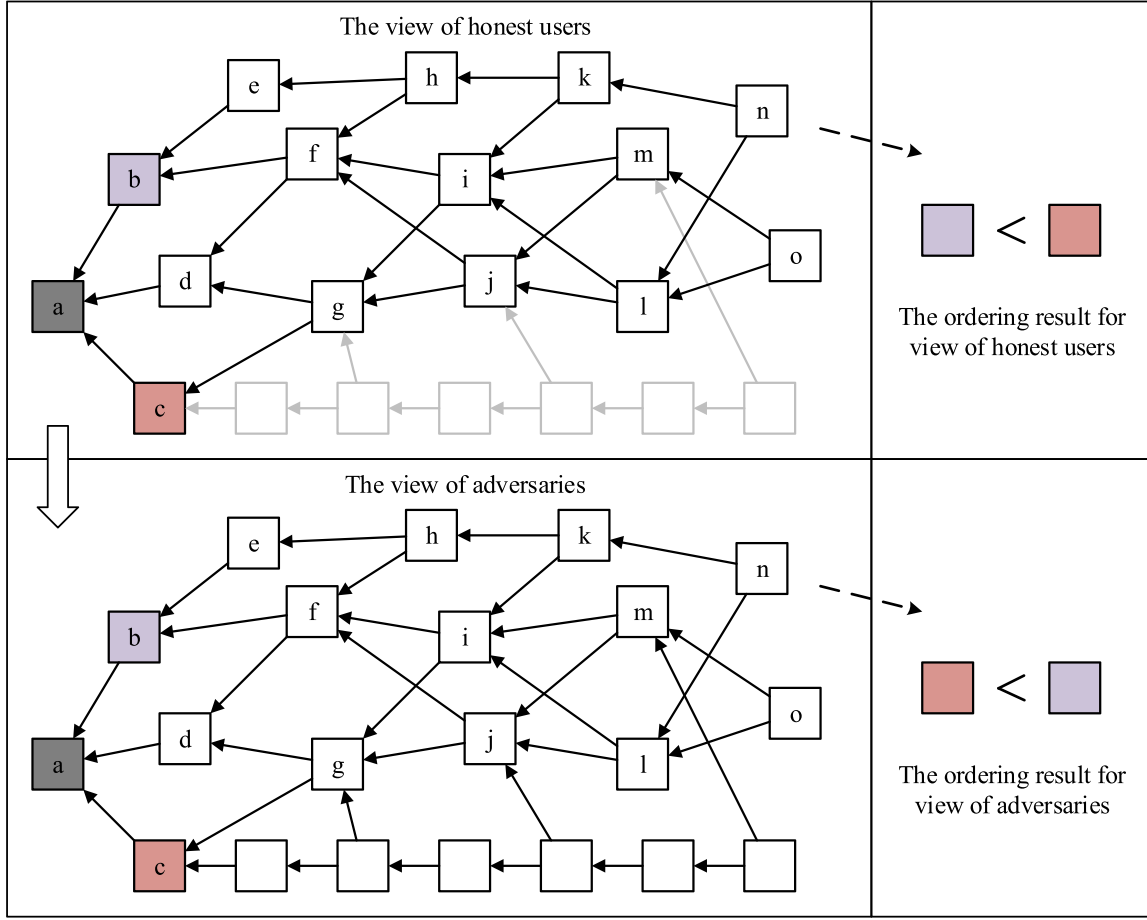


Fig. 3. Threat model.

$$\begin{aligned}
 &= \{D, E, F\}. \\
 \text{tips}(G) &= \{L, M\}.
 \end{aligned} \tag{3}$$

The system is adapted to distributed and dynamic networks, such as Internet of Things, consisting of edge nodes with certain computing power and communication bandwidth. Each node can solve a basic hash puzzle to generate a valid block. Adversaries can control parts of nodes with no more than half of the computing power of the entire networks. Due to the loose and unstable network topology, adversaries can only delay the message. We build the system under the assumption of synchronous networks, where there is a maximum transmission latency  $\Delta$  among all honest nodes, which means that if a node broadcasts a block message, others will receive it after waiting for  $\Delta$  time at most.

### B. Threat Model

Existing DAG-based blockchains ask a new block refer to arbitrary numbers of previous blocks, which puts as many blocks as possible to the graph, and thus significantly save computing power compared to chain-based blockchains. Then, those schemes present a consensus algorithm that inputs the current graph and outputs a subgraph consisting of a block set. According to reference relations and tie-breaking rules, the

algorithm can decide the global ordering of the BlockDAG. Unfortunately, such block reference does not promote consensus consistency but instead introduces potential attacks, because it is not enough to simply use graph topology to decide the ordering of blocks.

In the splitting attack we explored, adversaries can concentrate their computation power on constructing an alternative block using selfish mining strategy to withhold the blocks and release them at his choosing time, and ultimately destruct consensus consistency. This attack is effective for Phantom. We still use the example, which we have described in the introduction part, to illustrate this attack in Phantom and how it achieves the final result of block sorting reversal. Supposing that blocks  $a, b, c, \dots, m, n, o$  are all in the honest subgraph (this is reasonable because we can find suitable  $k$  to make these blocks all in the maximum  $k$ -cluster subgraph, and adversary only profits by launching attacks on honest subgraphs in Phantom). In  $G = (\mathcal{V}, \mathcal{E})$  as shown in Fig. 3, The adversary's target is the purple block  $b$ , attempting to make the red block  $c$  a conflicting block. From the view of honest users

$$\begin{aligned}
 \text{past}(b, G) &= \text{past}(c, G) = \{a\} \\
 \text{future}(b, G) &= \{e, f, h, i, j, k, m, l, n, o\} \\
 \text{future}(c, G) &= \{g, i, j, k, m, l, n, o\}.
 \end{aligned}$$

The connectivity of block  $b$  is higher than that of block  $c$ , so the purple block is prior to the red one according to Phantom's greedy algorithm.

However, the adversary conducts an attack beginning from the block in gray, issuing the block in red to combat the block in purple created by honest users. Specifically, the adversary secretly generates blocks to confirm the red block but does not publish them at the start. The blocks deliberately refer to the block created by honest users for including more blocks. The adversary incessantly generates blocks till there are more blocks to refer the red block than the purple block. From the view of the adversary, the number of blocks in set  $future(c, G)$  has increased by 6 and the number of blocks in set  $future(b, G)$  has increased by 3, which means that  $8 + 6 = 14$  blocks directly or indirectly refer the red block. Compared to the only  $10 + 3 = 13$  blocks in set  $future(b, G)$ , the connectivity of  $c$  is already higher than that of  $b$ , so the red block is prior to the purple one. The adversary then publishes all the secret blocks to change the order of the red and purple blocks in the system. Note that during this time period, the total blocks generated by the adversary is seven, and the total blocks generated by the honest miners are thirteen (block  $c$  is also generated by the adversary). This implies that the proportion of the computation power from the adversary is about  $7/(7 + 13) = 35\%$ , which means that the adversary only needs 35% of the entire network computing power to achieve the reversal of block order. This follows the assumption of the adversary capability in system model. That is to say, Phantom cannot ensure consensus consistency under splitting attacks. Our results in the simulation section also prove this point.

### C. Observations and Principles

The strategy to defend against splitting attacks is to maintain the stability of blocks' order. The first goal is to design a mining protocol so that honest miners no longer randomly select as many previous blocks as possible during mining. The second goal is to develop suitable block referencing strategies that enables miners to mine on the basis of suitable previous blocks, thereby concentrating computing power in the correct direction. We need to achieve the mentioned goals, so that it can effectively adjust the DAG topology to enhance the stability of ordering results.

We hope to obtain valuable information for achieving the above goals on the BlockDAG by analyzing the state-of-art schemes. Conflux [27] assigns the adaptive weight for blocks to decide the pivot chain in the graph, where the blocks with a lower hash value get a higher weight value (note that preferring choosing blocks with smaller hashes in consensus protocols may give the attackers more information to gamble the system and thus weakens the security [39]). Spectre [18] treats reference relations as votes to decide the pairwise ordering of blocks, where the ambiguous blocks just follow the majority of votes to amplify the gap. We have the following observations.

*Observation 1:* The block with higher difficulty in mining mainly affects the result of consensus.

Since the honest miners have the majority of computing power, most of the existing works usually put the blocks with

higher difficulty to be sorted before those with lower difficulty. Thus, if block  $B$  has higher difficulty than block  $D$  in the aforementioned example, then block  $B$  is sorted before block  $D$ .

*Observation 2:* The block referring to multiple parallel blocks in the BlockDAG has no substantial contribution to deciding the order of those parallel blocks.

Besides topological order, confirmation depth can also be used to order blocks. Here, the depth of a block is defined as the number of its reachable blocks. We still take the aforementioned example to demonstrate confirmation depth. Since block  $E$  refers to block  $B$ , the depth of block  $B$  increases 1. Block  $E$  also refers to block  $D$ , so does the depth of block  $D$ . Therefore, block  $E$  has no substantial effect to decide the order of block  $B$  and block  $D$ . Based on the observations, we explore two principles to produce a stable consensus protocol for parallel blocks as follows.

*Principle 1:* The blocks with higher difficulty are advised to refer to more blocks.

*Principle 2:* The blocks that can decide the order of the parallel blocks have high-priority to be referred to.

Next, we use both principles to design a mining protocol and two block reference strategies. The former decides the number of block references, while the latter affects the quality of block references.

## V. THE PHANTASM PROTOCOL

In this section, we present a new consensus protocol named Phantasm. It inherits the ordering protocol of Phantom but retrofits the mining protocol, making it adaptive and scalable. The mining protocol reconstructs the algorithm for selecting previous blocks by implementing an adaptive scalable mining method, allowing for setting reference priority strategies to affect the DAG topology.

### A. Adaptive Scalable Mining

The key to controlling the number of block references is that deferring the decision of block references in the mining phase. The classic way is to select the referred blocks and compute the current block puzzle. By constructing a Merkle tree of those referred blocks, we use the root hash as the reference of the block to compute the puzzle. Once a valid block is mined, the actual references are appended in the form of Merkle proofs to the block and being broadcast.

In our protocol, we modify the block structure and present an adaptive scalable mining function. The data field of previous\_hashes in the block header is changed to refs\_root (an element), which is used to store the root of Merkle tree built by all candidate block references. A new data field of references (an array) is added to the block body, in which each element includes data fields of hash (an element) and proof (an array) to form a Merkle path for the selected block reference. The pseudo-code of the adaptive scalable mining is shown in Algorithm 1.

In lines 3–4, the algorithm uses the hashes of all tips  $bs$  in the current BlockDAG to build a Merkle tree  $M_{bs}$ . line 7 asks block  $B$  to refer to the root of  $M_{bs}$  instead of the hashes of  $bs$ . Next,

**Algorithm 1:** The Adaptive Scalable Mining in Phantasm.

---

**Input:**  $G$  - the current BlockDAG  
**Output:**  $B$  - a valid block or null

```

1: function Mining( $G$ )
2:   Get the target  $T$  and difficulty  $D$ ;
3:    $bs \leftarrow tips(G)$ ;
4:    $M_{bs} \leftarrow$  Merkle tree of the hashes of  $bs$ ;
5:   // process the block header
6:   Construct a block and fill in the necessary fields;
7:    $B.refs\_root \leftarrow$  root of Merkle tree  $M_{bs}$ ;
8:    $B.nonce \leftarrow new\_nonce()$ ;
9:   // try nonce and compute the block difficulty
10:   $h \leftarrow hash(B)$ ;
11:   $d \leftarrow T/h$ ;
12:  if ( $d > D$ ) {
13:    // process the block body
14:     $m \leftarrow floor(d/D)$ ;
15:     $rbs \leftarrow RankBlocks(bs, m)$ ;
16:    for ( $i = 0; i < len(rbs); i++$ ) {
17:       $t \leftarrow hash(rbs[i])$ ;
18:       $B.references[i].hash \leftarrow t$ ;
19:       $B.references[i].proof \leftarrow M_{bs}.MerkleProof(t)$ ;
20:    }
21:    return  $B$ ;
22:  }
23: return null;

```

---

we follow a similar program logic to try nonces for the block puzzle. The block header simply contains the root of the Merkle tree of block references while their content can be specified after the algorithm finds a solution to the block puzzle so that it makes mining *scalable*. In lines 14–19, the algorithm evaluates the block difficulty and computes the permitted max number  $m$  of block references. The lower hash value the block is, the bigger the number  $m$  is. By doing so, we meet the principle 1 and make mining *adaptive* for block difficulty. Finally, the algorithm calls the RankBlocks() function to sort all tips and get no more than  $m$  blocks with higher priority. The corresponding Merkle path of these blocks is iteratively appended to block  $B$  and return it. The next subsection will show how we define the priority of the tips in the BlockDAG and sort for them.

### B. Block Reference Strategies

We design block reference strategies for the proposed mining protocol. The honest miners first solve the block puzzle and then adopt the block reference strategy to issue the mined block. By setting the priority for block reference selecting, the appended blocks affect the DAG topology in a positive way so that resist possible attacks. Here We give the following two strategies from the importance of the block.

1) *Max Cumulative Score*: BlockDAG consensus uses the heaviest subgraph to decide a consistent set of blocks. It can easily insert conflicting blocks or append malicious forks to the heaviest subgraph. We describe a possible attack as follows. First, the adversary chooses a goal block that is contained in

**Algorithm 2:** Ranking The Tips Using Cumulative Score.

---

**Input:**  $bs$  - the set of tips,  $m$  the max number of references  
**Output:**  $r$  - the list of ranked tips

```

1: function RankBlocks( $bs, m$ )
2:    $c \leftarrow$  sort tips  $bs$  by max cumulative score;
3:    $\alpha \leftarrow$  get the contribution parameter;
4:    $s \leftarrow 0$ ;
5:    $r \leftarrow []$ ;
6:   // filter the tips selected to be referred to
7:   for ( $i = 0; i < \min(len(c), m); i++$ ) {
8:      $t \leftarrow \log(\alpha + c[i].score/c[0].score) / \log(\alpha + 1)$ 
9:     if ( $t > 0$ ) {
10:       $s += c[i].score * t$ ;
11:       $r.append(c[i])$ ;
12:    }
13:  }
14:  // calculate the final score
15:   $s += len(r)$ ;
16:  update the final score of the mined block as  $s$ ;
17: return  $r$ ;

```

---

the heaviest subgraph. Then, the adversary creates a conflicting block without referring to the goal block. Next, it tries its best to make the heaviest subgraph contain the conflicting block. The attack is successful if the conflicting block is finally sorted before the goal block. The adversary can repeat this procedure until it succeeds. Through careful observations, we find the key reason: BlockDAG consensus does not reject the stale block in block references. A stale block in references refers to a block that is not newly created, but is still being referred to when creating new blocks.

An intuitive idea is to assign a score for each block and the new block accumulates the score of referred blocks. Thus, the final score of a block consists of the assigned score plus the cumulative score extended from the referred blocks. To avoid including the stale blocks, we let miners first refer to the blocks with a max cumulative score and define the score computation method. The closer the score of referred blocks, the larger the cumulative score. We use the logarithm function to compute the score contribution ratio and introduce a contribution parameter  $\alpha \in (0, 1]$  to control the reference range. For all tips sorted by cumulative score, let  $c_{\max}$  denote max score and  $c_i$  denote the score of block  $i$ . We compute the contribution ratio of each tip as follows.

$$r_i = \log(\alpha + c_i/c_{\max}) / \log(\alpha + 1). \quad (4)$$

where the larger the parameter  $\alpha$ , the larger the contributed score. We take the max score of referred blocks as the benchmark and each of them contributes a cumulative score in a certain proportion to the mined block, where the score is farther from the max score and the contribution is less. We denote the assigned score of the current block as  $a$  and thus the cumulative score is computed as follows.

$$s = a + c_i * r_i. \quad (5)$$



**Algorithm 3: Ranking The Tips Using Shortest Path.****Input:**  $bs$  - the set of tips,  $m$  the max number of references**Output:**  $r$  - the list of ranked tips

---

```

1: function RankBlocks( $bs, m$ )
2:    $c \leftarrow$  sort tips  $bs$  by the longest length of the shortest
   path to the genesis block in the BlockDAG;
3:   // calculate the permitted number of references
4:    $n \leftarrow \min(\text{len}(c), m)$ ;
5:    $r \leftarrow c[1:n]$ ;
6:   return  $r$ ;

```

---

According to principle 1, we treat the practical number of block references as the assigned score of the mined block, which means that the smallest score of a block is 1. The genesis block's score is set to 1 by default. We give the pseudo-code of tips ranking in Algorithm 2, where line 8 computes the score contribution ratio of each referred block. Lines 9–12 filter the useless references because the computed cumulative score is negative. Line 15 considers the number of referenced blocks as a contributing factor to the score for newly mined blocks by adding the length of  $r$ . Finally, we get the selected references for the mined block and update its score.

2) *Longest Shortest Path*: The presented method above seems like effectively resist the attacks, but it does not thoroughly refuse the stale blocks. Indeed, this strategy performs better than doing nothing, which reduces the probability of successful attacks (the results are shown in Section VII). The crucial factor is that it simply considers the score gap of block references but fails to pursue the newest mined blocks. As a result, the block with a large cumulative score still possibly contains stale blocks and is not in the heaviest subgraph. From this view, we ought to guide the mining in the direction of expanding the length of BlockDAG longer.

Compared to block height in the single-chain structure, it is more natural to introduce the notion of block length in the BlockDAG structure. Our intention is to find which fork contains stale blocks and recommend miners not to extend it. Thus, we define the length for blocks, which is the length of the shortest path in the graph from the current block to the genesis block. Dijkstra algorithm can optimally compute the shortest path for each block, which is beyond the scope of this paper. Here we just invoke this algorithm to get the length of the path in the BlockDAG.

We use the block length to rank all candidate tips, with which larger length has higher priority to refer to. Followed by principle 2, we show the pseudo-code of tips ranking in Algorithm 3. The RankBlocks() function inputs the current tips in the BlockDAG and the max number of references computed by block difficulty, and it outputs a list of tips ranked with the block length. Lines 4–5 calculate the number of references and cuts the ranked list to retain only that number. The algorithm finally returns the processed list.

The longest shortest path strategy guides miners to reference blocks with the longest shortest path, meaning that miners are

inclined to reference newer blocks rather than older or stale ones. In contrast, the heaviest subgraph strategy (as used in the Phantom protocol) allows a new block to reference all tip blocks without a particular bias towards the newest blocks. This gives an adversary the opportunity to create a conflicting block and by creating other blocks referring to this conflicting block directly or indirectly through stale blocks (i.e., increasing the connectivity of conflicting blocks to be greater than the connectivity of the attacked block), manage to include the conflicting block in the heaviest subgraph, thus influencing the block ordering. If the longest shortest path strategy is adopted, miners prefer to reference the newest blocks. This makes it difficult for the attacker to insert the conflicting block into the blockchain because the attacker needs to succeed in generating blocks in consecutive rounds to catch up with the progress of the honest miners. This increases the difficulty for the adversary to successfully execute a splitting attack, hence the longest shortest path strategy can more effectively resist splitting attacks. We give a formal proof in the next section.

## VI. SECURITY ANALYSIS

In the DAG-based blockchains, the mining protocol generates a BlockDAG while the ordering protocol makes consensus on it (involving block coloring and sorting stages). The results of consensus are directly affected by the topology of the BlockDAG, in which the adversaries can intentionally create some blocks to change it and reach the goal of double-spending attacks against a specified block. The attack succeeds if the specified block is sorted behind the conflicting block in the Blue set.

*Splitting Attack*: Adversaries insert a conflicting block to the honest subgraph and make a subgraph to let the conflicting block be sorted before an attacked block. Once both the conflicting block and the attacked block are in the Blue set while the block sorting stage decides the conflicting block is sorted in priority, adversaries publish the rest subgraph to make the attack successful.

Our protocol adopts the ordering protocol of Phantom to make consensus. However, the mining protocol of Phantom is weak against splitting attacks. The adversary can insert a conflicting block to the honest subgraph, in which each block can be treated as a vote to the block. We show the proposed consensus protocol Phantasm can adjust the structure of DAG to stabilize the ordering results, thus effectively reducing the probability of successful splitting attacks.

*Lemma 1*: The block that refers to all tips is equivalent to giving a useless vote on block coloring.

*Proof*: Given a BlockDAG  $G$ , for any block  $B \in \text{tips}(G)$ , a candidate block to be referred to, miners want to append a new block  $B'$  to  $G$ , so block  $B'$  is absolutely not in the  $\text{past}(B, G)$ . Due to block  $B'$  does not refer to block  $B$ , block  $B'$  is also not in the  $\text{future}(B, G)$ . According to the definition of anticone, block  $B'$  is in the  $\text{anticone}(B, G)$  and also block  $B$  is in the  $\text{anticone}(B', G)$ .

The ordering protocol uses  $k$ -cluster to color the blocks on a BlockDAG  $G$ . If the size of a block's anticone is larger than  $k$ , then the block is excluded from the Blue set. We assume a



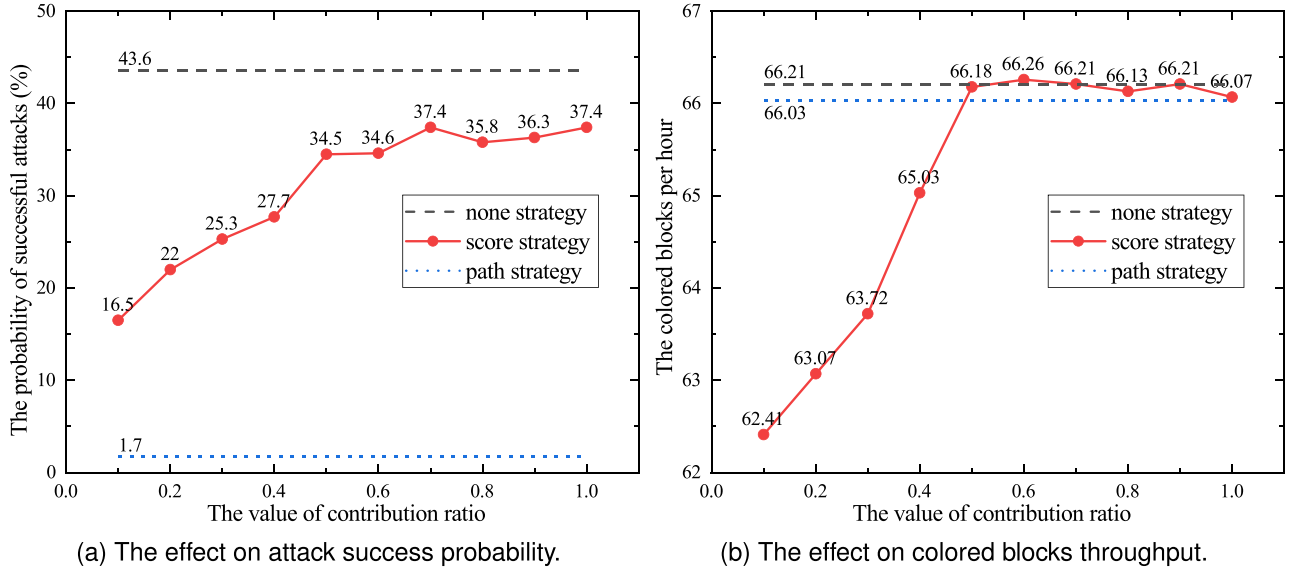


Fig. 4. Select different contribution ratios  $\alpha \in (0, 1]$  to compare the max cumulative score strategy with none strategy of phantom and longest shortest path strategy of Phantasm. For clearly, the key parameters are default set as follows: blocks per minute  $b = 0.5$ , cluster factor  $k = 10$ , and confirmation depth  $d = 100$ . The throughput is computed by simulation time with the hour unit. The average result is got from 1000 simulations.

Blue set  $U$  and a Red set  $V$ , and a new appended block  $B'$  will change the coloring results according to its references. If block  $B'$  only refers to the tips in the set  $U$ , then blocks in the set  $V$  increase their anticone by one. It means that block  $B'$  votes to support or consolidate the set  $U$  being colored blue. Conversely, blocks in the set  $U$  increase their anticone by one, which also increases the possibility of the set  $U$  being colored red. When block  $B'$  refers to all tips both in the set  $U$  and  $V$ , neither of the sets needs to increase the anticone. Block  $B'$  cannot change the advantage of either side being colored blue, and thus giving a useless vote on block coloring.  $\square$

**Theorem 2. (stable consensus):** Phantasm consensus protocol is more stable against splitting attacks.

**Proof:** Phantasm determines the priority of the blocks due to their depths. Referencing high priority blocks when creating new block stabilizes our consensus against splitting attacks. Consider the general situation of splitting attacks: adversary inserts a conflicting block  $B'$  to the honest subgraph without referring to the attacked block  $B$ , creates other blocks to refer to block  $B'$ , and attracts the blocks in the Blue set to refer to that. The mining protocol of Phantom lets an honest block refer to all tips. By Lemma 1, the honest block is useless to change the block coloring, but the malicious block can intentionally choose tips so that affects the block coloring. The reference strategy provides an ability to combat such malicious behavior, which guides the honest blocks to refer to the tips with a larger block length. Since the malicious blocks have to contain the conflicting block that has a shorter block length, the reference strategy adopted by honest miners can resist inserting conflicting blocks to make consensus stable against splitting attacks. We attempt to give the formal analysis for the probability of successful attack below.

We start by making the assumption that an adversary controls a fraction  $a \in [0, 1]$  of the entire network's computational power.

Typically, this value  $a$  is less than 0.5, which implies that the majority of miners are honest. We also incorporate two important parameters into our model: network delay  $\Delta$  and block difficulty  $d$ . These parameters correspondingly introduce probabilities  $\rho_\Delta$  and  $\rho_d$ . The parameter  $\rho_\Delta$  indicates the probability of the adversary creating the necessary delay to initiate a splitting attack. An increased  $\Delta$  leads to a larger  $\rho_\Delta$ , thereby facilitating the occurrence of a splitting attack. Conversely, a network without delay can effectively neutralize such an attack.

The parameter  $\rho_d$  represents the probability that the adversary generates a conflicting block with a difficulty surpassing that of the attacked block, which has a difficulty of  $d$ . Hence, a larger  $d$  results in a smaller  $\rho_d$ .

The adversary's objective is to position its conflicting block ahead of the attacked block in the blockchain. If the adversary calculates a conflicting block with a difficulty higher than the attacked block, it can simply wait until the conflicting block receives sufficient confirmations, with probability denoted as  $\rho_d$ . However, if this is not the case, the adversary needs to garner at least one additional vote block for the conflicting block (but not the attacked block), and the block with the extra vote must be colored blue. The probability in this case can be denoted as  $(1 - \rho_d) * P_{strategy}$ . Here,  $P_{strategy}$  represents the probability that the adversary creates an effective vote. We can express the adversary's success probability as follows:

$$\Pr_{Adv} = \rho_\Delta * (a * (\rho_d + (1 - \rho_d) * P_{strategy})) . \quad (6)$$

Due to the  $k$ -cluster constraint, the adversary must ensure that the additional block has fewer than  $k$  anticone blocks. As per Phantom's block reference rule, the probability of  $P_{strategy}$  in this case, which we denote as  $P_{all}$  is computed as follows, where  $(1 - a)^i a$  denotes the probability that the adversary successfully

creates an additional block colored in blue given that the additional block already has  $i$  anticone blocks created by honest miners. We only calculate up to  $k-3$  as the conflicting block and the additional vote block have already been accounted for

$$P_{all} = a + (1-a)a + \dots + (1-a)^{k-3}a$$

$$= \sum_{i=0}^{k-3} (1-a)^i a. \quad (7)$$

For that in our protocol, only the tips with a similar block length can be selected. The adversary has to create more extra blocks (i.e.,  $a^i$  extra blocks) to chase the honest subgraph. Thus, we have the probability of  $P_{strategy}$  in this case, which we denote as  $P_{len}$  as follows,

$$P_{len} = a + (1-a)a^2 + \dots + (1-a)^{k-3}a^{k-2}$$

$$= \sum_{i=0}^{k-3} (1-a)^i a^{i+1}. \quad (8)$$

Except for the first item, each item of  $P_{len}$  is less than  $P_{all}$  because the value of  $a$  is less than 0.5.

$$P_{len}.item[i]/P_{all}.item[i] = a^i < 1. \quad (9)$$

Thus, we can get the result of  $P_{len} < P_{all}$ . Finally, we prove that the designed reference strategy can make consensus more stable against splitting attacks.  $\square$

## VII. SIMULATION VERIFICATION

We implement the Phantasm protocol based on the Github repository [40] provided by phantom's authors, applying the presented block reference strategies to block generation. The source code is written in Python language, which uses the SimPy package (a process-based discrete-event simulation framework) to generate a peer-to-peer network, and is open sourced<sup>1</sup> on GitHub. Miners are partially connected and broadcast new blocks to expand the BlockDAG, using discrete-time to simulate the time consumption of network latency and block generation. We modify the program logic in the mining procedure, using the Poisson function to get the maximum number of block references. According to the two strategies (Section V), the miners first refer to blocks with a higher priority.

The experiments are conducted on a laptop that runs Windows 10 with AMD Ryzen 7 5800H CPU and 16 GB RAM, getting the average results from 1,000 simulations. The simulation gives a default configuration to execute the program where the relevant parameter setting is shown in Table I. It sets 5 honest miners and a malicious miner who owns 45% of the total computing power of all honest miners. The honest miners randomly choose 2 miners as their neighbors and the malicious miner makes full connections, where the communication delay is a Poisson random distribution with the intensity of 30 seconds. Some key variables influence the experimental results, such as blocks per minute  $b$ , cluster factor  $k$ , confirmation depth  $d$ , and contribution

TABLE I  
PARAMETER SETTING IN THE SIMULATION

Parameter	Setting
Number of honest miners	5
Malicious hash rate	45%
Propagation delay	30
Blocks per minute $b$	[0.5, 1.5]
Cluster factor $k$	[5, 15]
Confirmation depth $d$	[50, 150]
Contribution ratio $\alpha$	(0, 1]

ratio  $\alpha$  in the max cumulative score strategy. We give the value range of these parameters.

*The Choice of  $\alpha$ :* Before comparing the proposed Phantasm with the phantom, we need to determine the contribution ratio  $\alpha$  in the max cumulative score strategy of Phantasm. Our goal is to find an appropriate value of  $\alpha$  that reduces the attack success probability as large as possible while maintaining the considerable colored blocks throughput, i.e., valid ordered blocks per unit simulation time. For simplification, we named the mining protocol of phantom as none strategy, while in the Phantasm the max cumulative score strategy is named score strategy and the longest shortest path strategy is named path strategy.

As shown in Fig. 4, we adopt different  $\alpha$  from 0.1 to 1.0 to conduct simulations of score strategy, getting the results of attack success probability and colored blocks throughput. The simulation parameters are set as  $b = 0.5$ ,  $k = 10$ , and  $d = 100$ , where the parameter  $b$  means a time gap between two consecutive mined blocks. Due to the propagation delay, the smaller value of  $b$  causes a higher attack success probability and a larger blocks throughput. Thus, we adopt a small value of  $b$  is to accelerate block generation for more clear results. We use those parameters to conduct the same simulations of none strategy (original Phantom protocol) and path strategy, of which the results are treated as the benchmark to evaluate the value of  $\alpha$  in score strategy.

Fig. 4(a) shows the effect on attack success probability for different values of contribution ratio, where the results are displayed in the label. The none strategy is pictured with the dashed line in gray while the path strategy is pictured with the dotted line in blue. Roughly seeing, the smaller the value of  $\alpha$ , the better effect to resist splitting attacks. The essential reason is that the smaller  $\alpha$  restricts miners from referring to blocks with a large score gap, and thus it may cause the system to include less valid blocks. We illustrate the effect on colored blocks throughput in Fig. 4(b). When the value of  $\alpha$  is less than 0.5, the throughput rapidly decreases lower than the acceptable range between none strategy and path strategy. On the whole, the appropriate value of  $\alpha$  is set to 0.5.

*The Results Comparison:* After determining the parameter  $\alpha = 0.5$  in score strategy, we compare the ability of three strategies on resisting splitting attacks. As shown in Fig. 5, three key parameters influence the results. Obviously, applying score strategy performs better than none strategy, while using path strategy in the mining protocol can get the optimal effect. We

<sup>1</sup><https://github.com/BerserkRugal/Phantasm>

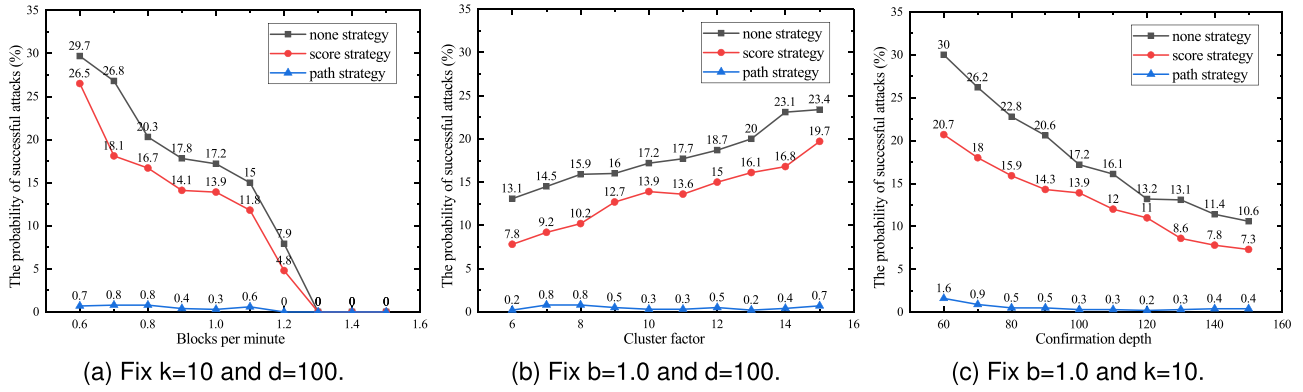


Fig. 5. The probability of successful splitting attacks under the mining protocol of phantom (none strategy in the block reference) and Phantasm (max cumulative score and longest shortest path strategies in the block reference). Three parameters are selected as control variables, which are blocks per minute  $b$ , cluster factor  $k$ , and confirmation depth  $d$ , to observe their influence on the probability. Each result is obtained from 1000 simulations and computes the average value.

adopt the method of control variables to understand the role of each parameter.

We control the variable as blocks per minute  $b$  and let the parameter  $k = 10$ ,  $d = 100$ . Fig. 5(a) shows sharply decrease in attack success probability, which is equal to 0 when  $b = 1.1$ . The large value of  $b$  means low speed in block generation. Be relative to the propagation delay, the network tends to be synchronous and thus effectively preventing the attacks. Fig. 5(b) sets the variable as cluster factor  $k$  while  $b = 1.0$ ,  $d = 100$ . As we can see, the larger value of  $k$  lets the adversary get more chance to insert a conflicting block, which causes a higher attack success probability. Finally, we control the variable as confirmation depth  $d$  while  $b = 1.0$ ,  $k = 10$  in Fig. 5(c). The longer time waiting for confirmation, the more difficult the adversary to conduct such attacks. Generally, we adjust this parameter to simply enhance security.

Note that, among the simulations with control variables, the attack success probability of path strategy is very low, where the results fluctuate under 1%. Such a strategy guide miners to expand the BlockDAG with the direction of longer length. As we analyzed, it can avoid referring to stale blocks and thus resist malicious behavior. The experimental simulation shows that our Phantasm protocol with the longest shortest path strategy indeed makes consensus more stable against splitting attacks than the phantom protocol.

## VIII. CONCLUSION

In this article, we have proposed a new DAG-based blockchain consensus protocol Phantasm to improve the stability of existing Phantom against splitting attacks. The core of Phantasm is a mining protocol, where the miners can adaptively and scalably choose tips (leaf blocks) in the DAG of blocks to refer previous blocks. By modifying the block structure, we allow miners to adjust the block references after obtaining a block puzzle solution, and the maximum number of the block's references is determined by its block difficulty. Subsequently, we design two block reference strategies (max cumulative score and longest shortest path) to guide the honest miners to refer to the blocks that have higher effects on the results of consensus. Through

appending blocks using such an adaptive scalable mining protocol, the DAG of blocks can effectively help the ordering protocol to distinguish the malicious blocks. The theoretical analysis and simulation experiments show that Phantasm with both two strategies performs stable against splitting attacks, where the longest shortest path strategy reaches optimal effect. We also observe in our experiments that the implementation of block reference strategy affects the throughput of colored blocks, as shown in Fig. 4(b). This is because miners are encouraged to reference the newest blocks, leading to a delay in block propagation. Therefore, we can further explore this issue.

## REFERENCES

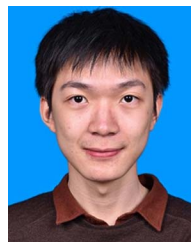
- [1] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin*, vol. 4, no. 2, 2008, Art. no. 21260. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [2] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [3] F. M. Benčić and I. P. Žarko, "Distributed ledger technology: Blockchain compared to directed acyclic graph," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 1569–1570.
- [4] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of DAG-based blockchain architectures," in *Proc. 12th Int. Conf. Open Source Syst. Technol.*, 2018, pp. 27–34.
- [5] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "SoK: Diving into DAG-based blockchain systems," 2020, *arXiv:2012.06128*.
- [6] W. F. Silvano and R. Marcelino, "IOTA tangle: A cryptocurrency to communicate Internet-of-Things data," *Future Gener. Comput. Syst.*, vol. 112, pp. 307–319, 2020.
- [7] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *Proc. IEEE 18th Int. Conf. e-Health Netw. Appl. Serv.*, 2016, pp. 1–3.
- [8] W. Yang, X. Dai, J. Xiao, and H. Jin, "LDV: A lightweight DAG-based blockchain for vehicular social networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5749–5759, Jun. 2020.
- [9] L. Li, D. Huang, and C. Zhang, "An efficient DAG blockchain architecture for IoT," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1286–1296, Jan. 2023.
- [10] F. A. Aponte-Novoa, A. L. S. Orozco, R. Villanueva-Polanco, and P. Wightman, "The 51% attack on blockchains: A mining behavior study," *IEEE Access*, vol. 9, pp. 140549–140564, 2021.
- [11] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting nakamoto consensus in asynchronous networks: A comprehensive analysis of bitcoin safety and chainquality," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 988–1005.
- [12] A. Averin and O. Averina, "Review of blockchain technology vulnerabilities and blockchain-system attacks," in *Proc. Int. Multi-Conf. Ind. Eng. Modern Technol.*, 2019, pp. 1–6.



- [13] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 906–917.
- [14] L. Kovalchuk, M. Rodinko, and R. Oliynykov, "Upper bound probability of double spend attack on spectre," in *Proc. 3rd Workshop Cryptocurrencies Blockchains Distrib. Syst.*, 2020, pp. 18–22.
- [15] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Comput. Sci. Rev.*, vol. 13, pp. 1–38, 2014.
- [16] G. Bu, Ö. Gürcan, and M. Potop-Butucaru, "G-IOTA: Fair and confidence aware tangle," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 644–649.
- [17] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 51% attack," *Appl. Sci.*, vol. 9, no. 9, 2019, Art. no. 1788. [Online]. Available: <https://www.mdpi.com/2076-3417/9/9/1788>
- [18] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," *Cryptol. ePrint Arch.*, 2016. [Online]. Available: <https://eprint.iacr.org/2016/1159>
- [19] Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM and GhostDAG: A scalable generalization of nakamoto consensus," *Cryptol. ePrint Arch.*, 2018. [Online]. Available: <https://eprint.iacr.org/2018/104>
- [20] Z. Zhang et al., "HCA: Hashchain-based consensus acceleration via re-voting," *IEEE Trans. Dependable Secure Comput.*, early access, 2023, doi: [10.1109/TDSC.2023.3262283](https://doi.org/10.1109/TDSC.2023.3262283).
- [21] Y. Liu et al., "VRepChain: A decentralized and privacy-preserving reputation system for social internet of vehicles based on blockchain," *IEEE Trans. Veh. Technol.*, vol. 71, no. 12, pp. 13242–13253, Dec. 2022.
- [22] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing. Fast money grows on trees, not chains," *Cryptol. ePrint Arch.*, 2013. [Online]. Available: <https://eprint.iacr.org/2013/881>
- [23] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Springer, 2015, pp. 507–527.
- [24] Y. Liu, W. Yu, Z. Ai, G. Xu, L. Zhao, and Z. Tian, "A blockchain-empowered federated learning in healthcare-based cyber physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2685–2696, Sep./Oct. 2023.
- [25] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Springer, 2015, pp. 528–547.
- [26] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," 2016, Accessed: Feb. 4, 2022. [Online]. Available: <https://byteball.org/Byteball.pdf>
- [27] C. Li et al., "A decentralized blockchain with high throughput and fast confirmation," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 515–528.
- [28] Z. Yin et al., "StreamNet: A DAG system with streaming graph computing," in *Proc. Future Technol. Conf.*, Springer, 2020, pp. 499–522.
- [29] L. Baird, "The Swirls hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," Swirls, Tech. Rep. SWIRLDS-TR-2016–01, vol. 34, pp. 9–11, 2016.
- [30] L. Baird, M. Harmon, and P. Madsen, "Hedera: A governing council & public hashgraph network," *The Trust Layer Internet, Whitepaper*, vol. 1, pp. 1–97, 2018.
- [31] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," 2018, Accessed: Feb. 4, 2022. [Online]. Available: <https://media.abnnewswire.net/media/en/docs/91948-whitepaper.pdf>
- [32] G. Danezis and D. Hryczyszyn, "Blockmania: From block DAGs to consensus," 2018, *arXiv:1809.01620*.
- [33] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 585–602.
- [34] H. Yu, I. Nikolić, R. Hou, and P. Saxena, "OHIE: Blockchain scaling made simple," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 90–105.
- [35] J. He, G. Wang, G. Zhang, and J. Zhang, "Consensus mechanism design based on structured directed acyclic graphs," *Blockchain: Res. Appl.*, vol. 2, 2021, Art. no. 100011.
- [36] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, 2018.
- [37] X. Boyen, C. Carr, and T. Haines, "Graphchain: A blockchain-free scalable decentralised ledger," in *Proc. 2nd ACM Workshop Blockchains, Cryptocurrencies, Contracts*, 2018, pp. 21–33.
- [38] I. Bentov, P. Hubáček, T. Moran, and A. Nadler, "Tortoise and hares consensus: The meshcash framework for incentive-compatible, scalable cryptocurrencies," in *Proc. Int. Symp. Cyber Secur. Cryptogr. Mach. Learn.*, Springer, 2021, pp. 114–127.
- [39] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 175–192.
- [40] Aviv Yaish, "Phantom (ghostdag)," 2021, Accessed: Feb. 4, 2022. [Online]. Available: <https://github.com/AvivYaish/PHANTOM>



**Zijian Zhang** (Member, IEEE) is an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is also a research fellow with the School of Computer Science, University of Auckland. He was a visiting scholar with the Computer Science and Engineering Department of the State University of New York at Buffalo, in 2015. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.



**Xuyang Liu** (Member, IEEE) received the BE degree in computer science and technology from the Beijing Institute of Technology, in 2022. He is currently working toward the PhD degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include applied cryptography, blockchain technology, distributed consensus.

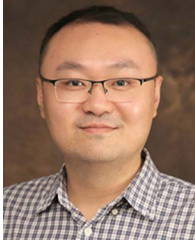


**Kaiyu Feng** (Member, IEEE) received the bachelor of engineering degree in computer science and technology from the Beijing Institute of Technology. He is currently working toward the master's degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His areas of research include blockchain technology, distributed systems, and databases.



**Mingchao Wan** (Member, IEEE) is the chief architect of Blockchain with the Beijing Academy of Blockchain and Edge Computing. He is also a member of the Beijing Nova Program, 2021. Prior to his current role, he served as a research scientist with IBM China Research Lab. His research interests include blockchain architecture, consensus and zero-knowledge proof.





**Meng Li** (Senior Member, IEEE) received the PhD degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He is an associate professor and dean assistant with the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He is also a post-doc researcher with the Department of Mathematics and HIT Center, University of Padua, Italy, where he is with the Security and PRIVacy Through Zeal (SPRITZ) research group led by Prof. Mauro Conti (IEEE Fellow). He was sponsored by ERCIM ‘Alain Bensoussan’ Fellowship Programme (from 2020.10.1 to 2021.3.31) to conduct post-doc research supervised by Prof. Fabio Martinelli with CNR, Italy. He was sponsored by China Scholarship Council (CSC) (from 2017.9.1 to 2018.8.31) for joint PhD study supervised by Prof. Xiaodong Lin (IEEE Fellow) in the Broadband Communications Research (BBCR) Lab with the University of Waterloo and Wilfrid Laurier University, Canada. His research interests include security, privacy, fairness, applied cryptography, cloud computing, edge computing, blockchain, and vehicular networks. In this area, he has published more than 60 papers in international peer-reviewed transactions, journals and conferences, including *IEEE Transactions on Dependable and Secure Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Database Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Smart Grid*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Green Communications and Networking*, COMST, MobiCom, ICICS, SecureComm, TrustCom, and IPCCC. He is an associate editor of *IEEE Transactions on Information Forensics and Security* and *IEEE Transactions on Network and Service Management*.



**Jin Dong** (Member, IEEE) is the general director of the Beijing Academy of Blockchain and Edge Computing. He is also the general director of Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing. The team he led developed “ChainMaker”, the first hardware-software integrated blockchain system around the globe. He has been long dedicated in the research areas such as blockchain, artificial intelligence and low-power chip design.



**Liehuang Zhu** (Senior Member, IEEE) is a professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from Ministry of Education, P. R. China. His research interests include cryptographic algorithms and secure protocols, Internet of Things security, cloud computing security, Big Data privacy, mobile and Internet security, and trusted computing.