SoK: Diving into DAG-based Blockchain Systems

Qin Wang[†,§], Jiangshan Yu[‡], Shiping Chen[§], Yang Xiang[†]
† *Swinburne University of Technology, Melbourne, Australia.*
‡ *Monash University, Melbourne, Australia*
§ *CSIRO, Data61, Sydney, Australia*

ABSTRACT

Blockchain plays an important role in cryptocurrency markets and technology services. However, limitations on high latency and low scalability retard their adoptions and applications in classic designs. Reconstructed blockchain systems have been proposed to avoid the consumption of competitive transactions caused by linear sequenced blocks. These systems, instead, structure transactions/blocks in the form of Directed Acyclic Graph (DAG) and consequently re-build upper layer components including consensus, incentives, *etc.* The promise of DAG-based blockchain systems is to enable fast confirmation (complete transactions within million seconds) and high scalability (attach transactions in parallel) without significantly compromising security. However, this field still lacks systematic work that summarises the DAG technique. To bridge the gap, this Systematization of Knowledge (SoK) provides a comprehensive analysis of DAG-based blockchain systems. Through deconstructing open-sourced systems and reviewing academic researches, we conclude the main components and featured properties of systems, and provide the approach to establish a DAG. With this in hand, we analyze the security and performance of several leading systems, followed by discussions and comparisons with concurrent (scaling blockchain) techniques. We further identify open challenges to highlight the potentiality of DAG-based solutions and indicate their promising directions for future research.

KEYWORDS

DAG-based Blockchain, SoK, Performance

1 INTRODUCTION

Limitations. Blockchain becomes an emerging technology to realize the distributed ledgers[1]. The rising interest in blockchain has attracted extensive attention. Thanking to its great potential to tackle critical security and trust challenges in various distributed environments, blockchain technology enjoys rapid development [1][2] with derived topics evolving into a well-studied field [3][4] in both industry [5][6] and academia [7][8]. However, the great impact of blockchain promotes an influx of participants joining the game. Continuously increased traffic results in unavoidably catastrophic congestion due to the performance bottlenecks including *slow confirmation*, *low throughput* and *poor scalability*. Compared to most centralized systems, these factors cannot be easily improved in blockchain systems which require decentralization as the priority. Several studies raise the view of blockchain trilemma [9] and the trade-off [10], claiming that the decentralization, security, and scalability, cannot perfectly co-exist in a blockchain system. For example, one major hindrance to Nakamoto Consensus (NC) [1] is the reduced security with an increased block generation rate. The security of NC's *longest chain wins* rule requires that honest nodes should be aware of other's blocks soon after the block's production. If the block creation rate exceeds the propagation time, concurrent blocks increase the possibility of forks happening. To mitigate the bottleneck of performance, multiple approaches have been proposed from different angles.

Potential Solutions. These approaches include the methods on sharding technique [11][12], layer2 protocols [13], sidechain technique [14], heterogeneous structure [15], hybrid consensus solutions [16] and assisted techniques [17] such as modifying hard-coded parameters [18] and cross-chain technique [19]. Sharding splits pending transactions into smaller shards and makes them processed in parallel, but it is hard to achieve the consensus across shards due to the asynchronization. Cross-chain protocols help to partially solve the problem by establishing the channels between multiple shards (horizontal sharding [19]). As a sacrifice, these on-top protocols inevitably result in heavy and redundant systems, weakening efficiency and stability. Layer2 protocols enable participants to perform off-(main)chain transactions through private communication rather than broadcasting to the whole network. Together with such features, the challenge is how to properly and effectively guarantee the validity and consistency between off-chain and on-chain transactions. The technique of sidechain pegs the auxiliary chains to involve more transactions. Heterogeneous structure adds the new types of blocks to assign them with different tasks. Hybrid consensus solution combines multiple fundamental consensus mechanisms together (PoX+BFT [15][16][20]) to integrate their benefits. Modifying hard-coded parameters *e.g.* directly increases the volume of block from 1M to 8M in BCH [18]. However, these techniques are still based on the linear-based backbone protocol, limiting the scope of exploitation. Therefore, the radical modification – reconstructing chains from the underlying structure and topology, become an emerging solution.

DAG-based Approach. Linearly structured blockchain systems maintain all the transactions/blocks in one single chain. Concurrent transactions/blocks compete for one valid position each round. This design inevitably leads to slow confirmation due to competitive miners, conflicted transactions, and wasted computations. The performance of the system can only be artificially suppressed (*e.g.* adjust the confirm time) so that each block is fully attached before the next one's arrival. Very few orphan blocks can be involved in the system. To this end, aiming to enable more transactions simultaneously processed/confirmed motivates the emerge of DAG-based blockchain systems [21][22][23][24]. They structure the transactions/blocks

[1]In this paper, we regard "blockchain" as a general term covering types and models which are based on the same technology. For simplicity, we ignore the difference between "blockchain" and "distributed ledger". Also, we occasionally use "DAGs" to represent "DAG-based Blockchains" for short.

in the form of graph topology to underlyingly alter the actual operations. DAG-based systems can improve performance by requiring less communication, computation and storage overhead. However, the sources of existed open-sourced projects and solutions are disparate and disorganized. Various basic concepts in DAGs (*e.g.* vertex meaning, consensus approach, order sequence) are still confused, and potential challenges have neither been clearly identified. An organized and structured systematic overview is absent for newcomers. Several studies draw their attention to this field, but the works are either rough in summaries [25][26][27][28][29][30][31][32], superficial in analysis [17][33][34][35][36][37][38][39][40], or incomplete in evaluations [41][42][43][44].

Contributions. In this SoK, we attempt to consolidate the core knowledge of the structural shift in blockchain systems and review the state-of-the-art DAG-based blockchain systems with comprehensive mechanisms and properties. To drive future research, we also provide multifaceted discussions and comparisons with concurrent scaling techniques of blockchain. Furthermore, we summarise unsolved problems existed in current DAG systems, hoping to highlight the research challenges in this field. We start with a series of simple questions, aiming to provide brief guidance for readers.

> *Q1. What is the DAG-based blockchain.*
> *Q2. How do they structure the ledgers.*
> *Q3. How to run the consensus.*
> *Q4. What are the desired properties.*
> *Q5. How secure are they.*
> *Q6. What is the performance improvement.*
> *Q7. How is this different from other techniques.*
> *Q8. What are the research challenges.*

Framework. To understand the DAG-based blockchains, establishing a DAG system step by step is necessary. We provide our analysis framework as follows. This framework helps to clear the mist surrounding the DAG technique applied in blockchain systems and provide a comprehensive systematic overview of such systems.

- ◇ ***Overview of DAG.*** Firstly, we review sufficient studies surrounding DAG-based blockchain systems in Section 2, covering the topics on protocol design, analysis and discussion, evaluation and simulation, improvement, application, and indirectly related analysis.
- ◇ ***Identified types.*** Firstly, we define the model to mathematically describe the DAG-based systems in Section 3, which contains two elements in different dimensions. Then, we obtain six types of structures by combining each element together. Last, we collect and category existed studies under our identified types.
- ◇ ***Consensus.*** Thirdly, we focus on the consensus mechanism of each aforementioned system in Section 4. Sepcifically, we deconstruct the consensus mechanism into several components, assisting to understand how these systems operate. We further discuss the featured techniques to show their internal designs and potential applications to the systems.
- ◇ ***Analysis.*** Then, we analyse the properties in Section 5, securities in Section 6 and performance in Section 7. These sections enable a deep understanding of the design principles, system features, and unique properties. Readers can learn the full views of collected systems through these multi-dimensional analysis.
- ◇ ***Discussion.*** Additionally, we review concurrent techniques surrounding the DAG and clarify a series of open questions in Section 8. Detailed discussions help us learn more about current challenges and the potential future of this field.
- ◇ ***Summary.*** As a summary, our analysis framework has answered the listed questions in detail. Discussions on elements of the DAG model provide us a simple but complete classification of current DAG-based systems, answering *Q1* and *Q2*. Deconstructions of consensus mechanisms show us how current DAG-based systems operate with their unique designs, answering *Q3*. Analysis of the properties, securities, and performance enables us to understand *Q4*, *Q5*, and *Q6*. Discussions, comparisons, and challenges answer *Q7* and *Q8*. Surrounding details about the origination, progress, and application are provided in Section 8, 9.

Insights. Our work provides a roadmap for studying the applied-DAG systems. Based on our scrutinized analysis, we provide several insights in this part: *a)* DAG-based blockchains are still far away from commercial applications due to their incompatible designs, absence of standards, unreliable security, varied performance, and unfinished implementations. *b)* DAG structure has the potential to improve scalability and performance but inevitably sacrifices certain properties like consistency or finality. *c)* DAG-based systems vary from one to another. A uniformly formalized model can hardly cover all key points. Instead, a loose and informal model benefits a lot for better understanding. *d)* Although applying DAG structure to classic distributed systems and blockchain systems is quite challenging, many studies with their proposed systems have greatly progressed and improved. We believe more exciting new schemes/protocols/solutions will emerge shortly.

2 OVERVIEW OF DAG-BASED BLOCKCHAINS

This section provides a quick overview of existed studies on DAG systems. Three aspects are summarized as follows.

Origination and Evolution. Directed Acyclic Graph (DAG) represents a finite directed graph with no directed cycles in the mathematic and computer field. Due to the unique topology, it is frequently employed as a basic data structure and applied to various algorithm scenarios such as seeking the shortest path in navigation and data compression in storage. The concept was first introduced to mainstream blockchains by Sompolinsky *et al.* in the work of GHOST [21][45], aiming to address the concurrency problem by allowing transactions structured in trees. The improved version [46] was applied to Ethereum [2] as its core consensus mechanism. GHOST protocol and its variants enable all blocks to quote more than one parent block, and all types of references can be converted into rewards as incentives. Thus, the irreversibility of chains can also be strengthened by blocks that are off the chain. After that, a shift of granularity from block-level to transaction-level was noted by Lerner *et al.* in DAGCoin [47]. Transactions take over the tasks of blocks, directly confirming the pending transactions and maintain the order of sequence. Consequently, efficiency was leap due to the

abandonment of packaging and competing steps. IOTA [22] and ByteBall [23], inheriting the concept of blockless, were proposed with full open-sourced implementation and leads the markets till now. In the following times, several modifications and updates were added to DAGs. Spectre [24] aims to establish a system that enables concurrent block creation. Hashgraph [48] proposed a permissoned graph-based blockchain with the consensus inspired by BFT-style protocols [49]. Nano [50] proposed a so-called *block-lattice* structure to realize immediate and asynchronous processing by allowing users to maintain their own lightweight accounts. Conflux [51] brought blocks back to the system for the purpose of a total linear sequence. More cases will be discussed in later sections.

Analysis and Evaluations. Open-sourced and published works provide their basic design patterns. Related studies based on them are rooted in different views. Several works provide analysis by components, including topics on backbone properties [52][53][54], consensus design [29][55], random walk tip selection [56], witness selection [28], cryptographic hash algorithm [57], timestamps [58], signature scheme [59], memory model [60], pending probability [61], component design [62], fees [63] *etc.* While some studies are based on properties, covering fairness [26], consistency [64], anonymity [65][66][67], performance [44][35][44], practicability [41], robustness [42], stability [52], lightweight client [68], security [43][69][70][71][72][73][74][75], incentive and rewards [76], *etc.* Discussions [25][27][77][36][38] and comparisons [39][37] on selected systems (*e.g.* IOTA, Byteball, Hashgraph, Nano) have identified several features and challenges, but they were technically or theoretically insufficient. In-depth analysis employs different techniques and tools, such as establishing an analysis framework [26] and converting complicated environment into simulations [34][33][53][78]. Meanwhile, several types of formal analysis have also been proposed, including both brief models with qualitative arguments [79][45][45], formal model analysis [80][81][82] and formal model with mathematics induction [83]. Beyond that, a few of modified protocols [84][85] and improved protocols [86][87][88] are proposed for further exploitation.

Applications. Systems based on DAG mainly benefit distributed applications (DAPP) with high performance and low cost. Existed solutions could be roughly categorised by their combination layers. Directly integrating building blocks with the underlying network [89][90][91] helps to enjoy almost equal properties and advantages of DAG, but it requires expertise development skills and costly hardware devices. Establishing applications through official components (*e.g.* Qubic [92], MAM [93] in IOTA) is an alternative selection for developers [90][94][95]. It simplifies the processes of deployment and maintenance but scarifies flexibility and customization to a certain degree. Beyond that, various specific scenarios are considered, which includes the fields on Internet of Things [96][97], data management [98], vehicular applications[99][100][101], charging scheduling [102], M2M Communications [103][104], manufacturing [105], smart home infrastructure [91], precision agriculture [90], P2P energy trading [106], intelligent transport systems [89], air-quality data-monitoring System [107], smart transportation services [90][108], sensor node system [109], data quality management [110], E-health [94][95][111][112], smart grid [113], feedback control system [114], database platform [115], voting system [116][117][118], named data network [119], smart city [120] *etc.* It should be noted that proposed schemes cannot simply achieve multifaceted requirements. They need to employ additional techniques such as IPFS file system [121] for storage, zero-knowledge proof (ZKP) for privacy, access control for authentication [98], *etc.* Finally, an important fact based on aforementioned solutions is that, all combinations aiming to implement DAPPs are still staying at the stage of Proof of Concept (PoC), far away from practical realizations and commercial applications.

Guidelines. For an overview, we summarise related works to provide a full picture of DAG. Among citations in this paper, the topics that are frequently studied include three aspects: the protocol designs (including both projects in industry and schemes in academic); the analysis, comparison, and evaluations; and potential applications to real scenarios. A handful of studies provide improved suggestions or schemes, and several works focus on the techniques or analysis (indirectly) related to DAGs. Detailed distributions are shown as in Table.1.

Table 1: Guideline of Literature

Protocol Design	[21] [46] [45] [22] [62] [23] [51] [48] [122] [123] [124] [125] [29] [60] [126][127] [24] [128] [50] [129] [130] [131] [132] [133] [47] [134] [135] [136] [79] [137] [138] [139] [140] [141] [142] [40] [85] [143] [144] [145] [146] [147] [148] [149] [150] [151] [152] [153] [154] [55];
Analysis/Discussion	[155] [156] [157] [158] [52] [57] [31] [60] [54] [59] [58] [63] [61] [56] [64] [65] [71] [159] [43] [84] [44] [25] [27] [160] [36] [37] [38] [161] [83] [74] [75] [162];
Evaluation/Simulation	[157] [53] [61] [26] [28] [42] [35] [34] [33] [41] [78] [163] [164];
Attack/Defense	[69] [70] [73] [71] [78] [72] [165] [161];
Improvement	[68] [88] [166] [67] [65] [66] [86] [87] [167] [168] ;
Application/Cases	[77] [99] [101] [105] [98] [110] [109] [95] [94] [90] [108] [107] [106] [169] [97] [170] [171] [113] [89] [91] [96] [100] [114] [116] [117] [103] [104] [118] [111] [120] [112] [119] [102];
Related Discussion	[11] [15] [16] [19] [172] [3] [13] [173] [76] [174] [175] [32] [176]

3 MODELING

This section defines an informal DAG model and network model. Then, we classify current systems into six types accordingly.

3.1 From Classic Blockchain To DAG

Blockchain records activities in the form of transactions. Transactions are organized into a hierarchical structure as a block, and blocks are arranged in an irreversibly ordered sequence. New blocks can only be attached to the main chain which are strictly sequenced

by predecessors and successors. Although consensus mechanisms vary from system to system, classic blockchains base on the same linear-based chain structure. Transactions cannot be tampered or compromised due to the global view shared by all participants, integrity and traceability are consequently provided. Existed modeling of blockchain could be referred to [45][46][177][178].

The performance bottleneck of linear structure is mainly caused by consensus mechanisms. Specifically, a group of nodes compete for the right of block packaging though methods such as election for leader (BFT-style), solving puzzles (PoW), holding stakes (PoX), *etc.* Only winners are able to determine the validity and confirmation of transactions, while left transactions are pended in the pool or discarded off the chain. Adding blocks at the same time causes more conflicts. In contrast, DAG-based structure aims to enable multiple transactions confirmed in one round. Each unit (transactions/blocks/events) of the ledger could refer to more than one parent units, and also could be referenced by numerous subsequent units. This structural design supports concurrent operations. Multiple nodes can simultaneously add units to the ledger, thereby significantly improving the throughput.

3.2 DAG-based Model

A directed acyclic graph $\mathcal{G}$ consists of a point set $\mathcal{V}$ and an edge set $\mathcal{E}$. Firstly, each element in the point set corresponds to a *unit*. A unit can be instantiated as a transaction $\mathsf{Tx} \in \mathcal{T}$, a block $\mathsf{B} \in \mathcal{B}$, or an event $\mathsf{E} \in \mathcal{E}$ in protocols, where $\mathcal{T}, \mathcal{B}, \mathcal{E}$ represents the sets of elements. Secondly, the element in the edge set is a tuple (u, v), which represents the partial order relationship between two points u and v. The relationship, in most cases, indicates that one of the *unit* directly/indirectly references another unit. For instance, the notation $u \leftarrow v$ means v confirms/verifies/witnesses/sees u, where $\{u, v\} \in \mathcal{V}$. We define the DAG-based model with two properties as follows:

Definition 1 (DAG-based model)**.** *The DAG-based blockchain model is defined as follows:*

$$
\begin{aligned}
&\mathcal{G} = (\mathcal{E}, \mathcal{V})^{\dagger\ddagger}, \textit{ where} \\
&\mathcal{V} = \{u \,|\, u \in \mathcal{T} \cup \mathcal{B} \cup \mathcal{E}\}, \\
&\mathcal{E} = \{(u, v) \,|\, u \leftarrow v \wedge \{u, v\} \in \mathcal{V}\}. \\
&\dagger : \forall u \leftarrow v \nRightarrow v \leftarrow u; \\
&\ddagger : \textit{Assume that } u_i = \{u_1, u_2, ..., u_l\} \subseteq \mathcal{V}, \\
&\quad \forall i, j, ..., k \in [1, l-1],\ k >= ... >= j >= i, \quad \textit{if} \\
&\quad u_i \leftarrow u_j,\ u_j \leftarrow u_k, ...,\ \textit{then}, u_i \leftarrow u_k \textit{ does not exist.}
\end{aligned}
$$

Here, the operation "a|b" means a satisfies the condition of b. The notation "←" represents an action[2] *that happens in the network. The property means (i) unidirectional*†*: the references point in the same direction; (ii) acyclic*‡*: no loop exists in graph. The properties enable the units in the network appended-only and orderable.*

Key Parameters. We capture several qualitative metrics as the basic terminologies to describe the systems. In contrast, we omit the quantitative parameters (like confirmation time, transaction fees, propagation time *e.g.*). These parameters can hardly be estimated due to the absence of implementations.

- ***In/Out degree***, denoted as $(|In|, |Out|)$, describe the number of connections of each unit. The in-degree $|In|$ represents the number of a certain unit's successors, while the out-degree $|Out|$ shows the number of a unit's ancestors.
- ***Transaction model*** describes how to complete a transaction. Three model types have been identified. *UTXO* [1] stands for the unspent outputs. The transaction in *UTXO model* is atomic and unsplittable. Every operation has to be completed through these transactions. *Account model* [2] maintains a *balance* field in its data structure. Transactions are finished via the changes in the user's balance. The user modifies its data in the local view and then synchronizes it to the network. A complete token transferring in *pair model* [50] consists of two coupled transactions: a "send" transaction signed by the sender and a "receive" transaction signed by the receiver. These two transactions are required to occur within a negligible time interval.
- ***Confidence*** is a cumulative number that is used to show the confidence of a unit being verified by subsequent units both directly and indirectly. It also reflects the probability of a unit that will be selected in the next round. The parameter can be instantiated as different forms in the systems, such as the cumulative weight in IOTA [22] and the witness in Hashgraph [48], the score in Phantom [127]..
- ***Identifiers*** are used to identify the unit in the network. Each system has its customized parameters. These parameters include: *height* (h) represents the length of a path, where the path starts from genesis transaction to a specified transaction; *depth* (d) means the longest reverse-oriented path, starting from the tip transaction to a certain transaction; *chain index* (i) identifies the chain with an index number when multiple chains coexist in the network; *logic clock* (t) plays the role of capturing chronological and causal relationships.

3.3 Identified Types

The abstracted model describes the protocols in a theoretical and mathematical form. **We first specify the exact representations of units $\mathcal{V}$ and edges $\mathcal{E}$ in the model**. These elements define the structure and topology of a DAG-based system. We start our analysis framework from here.

- ***Unit Representation.*** This indirectly shows the underlying structure of a system, either transactions, events, or blocks. We have two types of options: 1^{od} and 2^{od}. The former represents the request that is immediately handled whenever it is received, without having to wait for more requests from peers. The forms of this type include transactions and triggering-events. The latter represents the request that requires further handling. In most cases, the requests in this type are pre-computed or packaged by powerful parties (like miner or validator) and then be disseminated. The forms of this type contain blocks and events.
- ***Network Topology.*** A single edge may be not enough to represent all relationships within a DAG-based system. The system is usually meshed by a set of edges. Three types of graph

[2] As aforementioned, the *action* can be instantiated as confirm/verify/witness/see in protocols. Noted that, the property of *transitive relation* is omitted in this model, because *action*s cannot be transmitted among units.

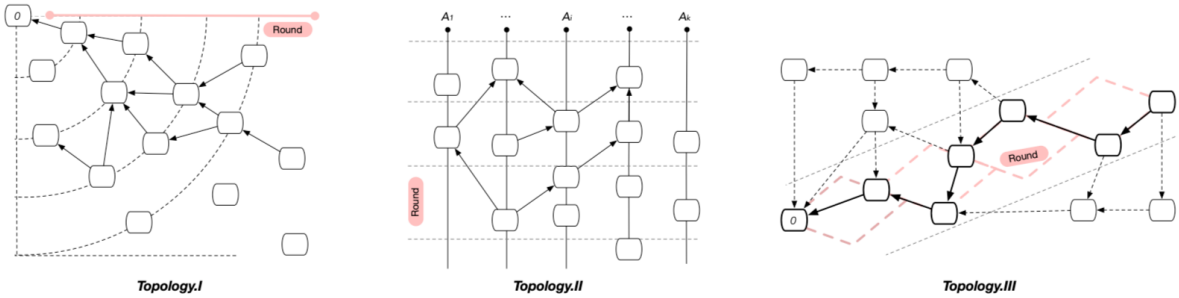

Figure 1: Topology Patterns

typologies are identified according to their trends (formed graphs), namely $\widehat{D}ivergence$, $\widehat{P}arallel$ and $\widehat{C}onvergence$, abbreviated as $\widehat{D}, \widehat{P}, \widehat{C}$, respectively. In detail, $\widehat{D}ivergence$ means the units sparsely spread in unpredictable directions without predetermined orders. $\widehat{P}arallel$ means the units are maintained in the form of multiple chains by a group of nodes. $\widehat{C}onvergence$ means the units are organized in a determined sequence or tend to converge in a determined sequence.

The *unit representation* indicates the structure of systems. It also determines the ledger execution model, indicating how a transaction is completed in DAG. We provide more details on transaction models. Generally, two types are included: UTXO-based model and account-based (*acct*) model. The first one means all operations have to be realized through atomic transactions. Users can calculate the balance by tracing the history of previous transactions. For the second one, each user holds an account and the transaction is configured as one of the fields in its structure. Users calculate the balance directly in their accounts. This is an important feature to understand the underlying structure of the DAG-related systems.

The *topology* mainly focuses on an eventual trend of these units in the network due to the uncertainty of unit confirmation in DAG systems. The units that initially spread to multiple directions but finally converge to the main chain, can still be classified into *convergence*. For example, the blocks in GHOST [45] attach to their parents in a disordered way, but they are eventually structured into the main chain. This metric indirectly shows the finality and consistency of a system.

Then, we identify six types based on the combinations of the options in each metric. We collect 30+ DAG-based blockchains systems (from 2011 to date) and classify these systems into the aforementioned types. Detailed classifications are shown in Table.2.

- $(\widehat{D}, 1^{od})$. For better understanding, we describe the situations with terms frequently used in blockchain. This type means the data structure is blockless and the topology is a natural graph. Transactions are in equal and fair entities. Exemplified systems include IOTA [22], Graphchain [79] and Avalanche [130]. We denote this type as *Type I*.
- $(\widehat{D}, 2^{od})$. Transactions needed to be organized in blocks for packaging, and the topology is a natural graph. Exemplified systems contains Spectre [24], Phantom [80] and Meshcash [81]. This type is denoted as *Type II*.
- $(\widehat{P}, 1^{od})$. Data structure is blockless, and transactions are maintained by individual nodes which forms multiple parallel chains. Exemplified systems cover Nano [50], Hashgraph [48], DLattice [170], Jointgraph [171], Chainweb [147], Aleph [167], Vite [139], Caper [179] and Lachesis-class protocols [122][123] [124][125][126]. This type is denoted as *Type III*.
- $(\widehat{P}, 2^{od})$. Transactions are structured in blocks to form the parallel chains. The systems include Prism [131], OHIE [145], Blockmania [132], Blockclique [180], Eunomia [146], Dexon [134] and PARSEC [149]. We denote this type as *Type IV*.
- $(\widehat{C}, 1^{od})$. Data structure is blockless, and transactions converge into a main chain. Examples are Byteball [23], Haootia [82] and JHdag [29]. The type is denoted as *Type V*.
- $(\widehat{C}, 2^{od})$. Transactions are Structured in blocks with a main chain. The systems include GHOST [45], Inclusive [46], CDAG [148], Conflux [51][140][141] and StreamNet[136]. We denote this type as *Type VI*.

Compared with Previous Classification. Previous classifications are coarse-grained that makes newcomers even researchers confused, such as the blocklattice mentioned in [134], blockDAG in [180], GHOSTDAG in [80], Lattice-based DAG in [15], *etc.* We observe that the gaps are due to the lack of classification metrics or dimensions. We, instead, classify existing systems in fine-grained forms by two dimensions (the topology made by edges, the units that indicate structures). With this in hand, we find that so-called TxDAG/blockDAG are named by their unit representations, while the terms GHOSTDAG/TreeDAG are based on their typologies. Our classification method, naturally, leads to the fact that each term in previous classifications maps more than one types in ours. For example, Lattice-based structure, or equally parallel-chains, covers both *Type III* and *Type IV* defined in our method.

4 CONSENSUS OF DAGS

This section aims to explain consensus in a simple way through the deconstruction. We present several key aspects of discussion in consensus mechanisms and provide the details of each system. Then, we further conclude featured techniques and give our discussions.

4.1 Deconstructed Components.

Achieving the consensus considers three aspects: who conduct the consensus, how do they operate, and what is the technique. We deconstruct the consensus into several decoupled components

Table 2: Identified Types with Corresponding Systems

	$\widehat{D}ivergence$	$\widehat{P}arallel$	$\widehat{C}onvergence$
1^{od}	IOTA[22], Graphchain[79], Avalanche[130] (*Type I*)	Nano[50], Hashgraph[48], DLattice[170], Jointgraph[171], Chainweb[147], Aleph[167], Vite[139], Caper[179], Lachesis[122][123][124][125][126] (*Type III*)	Byteball[23], Haootia[82], JHdag[29] (*Type V*)
2^{od}	Spectre[24], Phantom[80], Meshcash[81] (*Type II*)	Prism[131], OHIE[145], Blockmania[132], Blockclique[180], Eunomia[146], DEXON[134], PARSEC[149] (*Type IV*)	GHOST[45], Inclusive[46], CDAG[148], Conflux[51], StreamNet[136] (*Type VI*)

for a better understanding of different systems and capturing their commons. The nodes that can conduct the consensus make up a consensus group, called a committee. *Openness* indicates whether this committee is open to every node. *Membership selection* defines the rules of becoming a committee member. *Unit allocation*, *unit positioning*, *extension rule* and *conflict solving* determine the methods to reach the agreement. *Featured technique* highlights the distinguished techniques in each system. Detailed aspects of discussion are stated as follows.

- ***Openness*** indicates whether an arbitrary node can run the consensus algorithm without permission. Two types of selections are included, namely *permissionless* and *permissioned*. A permissionless DAG system means every node can join/leave the committee without restrictions. The size of the committee is dynamic. A permissioned DAG system means the new-coming nodes are required to obtain permission when joining the committee. The permitted conditions are unusually predefined by founder teams or core members. The size of the committee is fixed.
- ***Membership selection*** defines the rules that are used to select the nodes into the committee. For a permissionless system, every node can participate in the competition for producing units. Nodes are ranked for priority according to their owned resources. The resources includes computing work (Proof of Work, *PoW*), stake (Delegated Proof of Stake, *PoS*/DPoS), vote (voting for favorable nodes, *Elect*). The node that holds/obtains the most resources has the highest probability to win the competition. For a permissioned system, nodes have to prove the fact that they can meet the predefined rules. The committee membership is designated by their developing teams (*Assign*).
- ***Unit allocation*** is a prelude for consensus. Units in this step are required to be allocated with an identifier (*Role*) or a destination (*App*, *Chain*). The identifier refers to a unit that plays a specific role with responsibilities for some tasks. This case rarely happens in our collected systems. The destination indicates that a newly appended unit is uniquely allocated to a specific place, instead of broadcasting to the network. This design arranges the units into isolated zones in advance to reduce the potential conflicts. The option *App* refers to a unit that is allocated to different applications, whilst *Chain* represents a unit that is allocated (randomly or by some rules) to a certain chain among all chains.
- ***Unit Positioning*** is a way to locate the unit in the network. This step is essential to sort the units into a total linear order. In classic blockchain systems, locating a block can reference the parameter of *height* (h). However, locating the unit in DAG systems is hard. For the systems based on *Topology* $\widehat{D}$, unstructured units make it almost impossible to precisely locate the unit in the expanding network. Only when the units are sorted into a linear sequence and buried deep enough, a unique location in the form of height (h) is obtained. In contrast, for the systems based on *Topology* $\widehat{P}$, chains with unique indexes (i) are structured in parallel. A unit can be positioned via the orthogonal parameters (i, h). Similarly, for the systems based on *Topology* $\widehat{C}$, any unit can be roughly located by key blocks for the first step on the main chain, and then be precisely searched inside these blocks by sub-parameters like hash, timestamp.
- ***Extension rule*** empathizes how to extend the chains/graphs and break ties. Tie-breaking occurs when equivalent forks (subgraphs) compete for one winner, which is essential to maintain consistency by deleting the overlapped branches and reducing the data overburden. *Extension rule*s conclude the main consensus mechanisms. Feasible methods includes Nakamoto consensus (*NC*) by the longest chain, variant Nakamoto consensus (*variant-NC*) by the heaviest weighted sub-tree, asynchronous Byzantine agreement (*async-BA*), classic PBFT protocols (*PBFT*) and its variation (*smpl-PBFT*), tip selection algorithm (*TSA*), some special algorithms like greedy algorithm (*GA*), recursive traverse algorithm (*RTA*) and *sampling* algorithm. For the systems without explicit rules, we use *natural* to denote the case.
- ***Conflict solving*** presents a set of parameters that determine the priority of conflicting units. The parameters include three types: a) the confidence of each unit, instantiated by the forms of *weight*, *confidence*, *fee*, *score*, *fitness*; b) the random beacon like *Hash*; and c) a natural sequence, containing *logic clock*, *appearance* and *rank*. Sometimes, the decision is made by powerful authorities such as the leader and coordinator. We denoted them as the *trusted role*s (TRs). Additionally, conflict solving differs from tie-breaking in their scope of adoption. Tie-breaking happens at the level of (main) chains, whereas conflict solving impacts on every single unit and mostly happens in the sorting algorithm with the aim to order units in linearization. Note that, in several systems,

both tie-breaking and conflict solving are integrated with the extension rule. The consensus mechanism, thus, reaches the agreement in one-step.
- ***Featured technique*** provides the techniques that are different with other systems. We highlight one main feature from each system to distinguish them from peers.

4.2 Consensus of Each Type.

This subsection provides our reviews on consensus mechanisms of current DAGs. We mainly analyse them in the view of our identified types, which helps to find some insights.

Consensus on Type I. Type I systems are blockless. The topology of transactions is a natural expanding graph. This type of systems include IOTA [82], Graphchain [79] and Avalanche [130].

IOTA [22] is a permissionless network where each node can freely participate and leave it. IOTA adopts the UTXO model as the data structure. This design makes IOTA establish the system through transactions. The transactions issued by nodes constitute the site set of the *tangle*, which is the ledger for an up-to-date history of transactions. All nodes in an IOTA network store a copy of the tangle and reach the consensus on its contents. Specifically, the extension of the tangle follows the rule[3] where one tip (pending/unapproved transaction) is required to approve two ancestor transactions. Thus, users who issue a transaction will contribute to the security of the system. However, as tips are continuously generated and attached to the tangle, formed sub-graphs inevitably spread in different directions. To prevent the network from split into isolated cliques, *tip selection* algorithms are essential for stability. The algorithms help to maintain a uni-direction graph by controlling the way to select tips. Three types of mechanisms are provided in [22]: *uniform random*, *unweighted random walk* and *weighted random walk*. All these mechanisms are based on statistical probability to simulate real scenarios. The most advanced mechanism is a weighted random walk algorithm, which is an application of Markov Chain Monte Carlo (MCMC) algorithms. MCMC could be transformed into other strategies through α, a configurable parameter used to control the effectiveness. When α converge towards 0, tip selection becomes uniformly random; while towards 1, tip selection becomes deterministic. Additionally, we introduce two types of modified tip selections as the improvements:

- *G-IOTA* [86] modifies the rule (the weighted random walk tip selection rule) that a tip can approve three ancestor transactions at one time, instead of two. The extra one selects a left-behind tip in the tangle, used to increase the fairness in terms of transaction confidence for all honest transactions. The algorithm allows left behind tips to regain the opportunity to be approved by incoming tips. G-IOTA further discusses the incentive mechanism to punish conflicting transactions and introduces a mutual supervision mechanism to reduce the benefits of speculative and lazy behaviors.
- *E-IOTA* [87] propose a parameterized algorithm to provide the randomness for tip selection. The algorithm sets two specific parameters p_1, p_2 where $0 < p_1 < p_2 < 1$. When appended to the network, a tip generates a random number r where $r \in (0, 1]$ to decide which types of mechanisms to conduct: $r \in (0, p_1)$ – uniformly random selection; $r \in [p_1, p_2)$ – low α weighted selection; or $r \in [p_2, 1)$ – high α weighted selection. Thus, E-IOTA controls the distribution of different types of mechanisms by changing the parameters p_1, p_2. This assists to establish a self-adjusted tangle. The algorithm suits for both IOTA and G-IOTA.

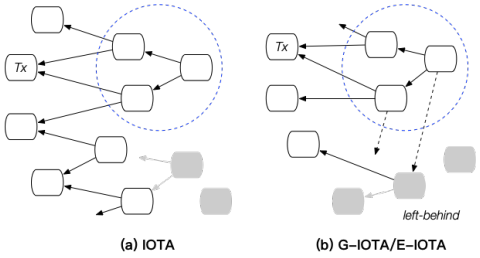

Figure 2: IOTA with its Improvements

Graphchain [79] is a permissionless network which has a similar design to IOTA. Graphchain is formed by tasking each transaction to confirm its ancestors. Specifically, a transaction must verify several (at least two) ancestor transactions and each transaction carries a PoW of elective difficulty. PoW cumulatively and transitively affirms the prior valid history. In contrast, Graphchain differs from IOTA in its incentive mechanism. IOTA removes incentives from participated nodes where all transactions are feeless. The system progresses under the tip selection rules and free from the influence of incentives. Graphchain, instead, introduces an incentive mechanism to maintain the graph. Transactions are required to post transaction fees as the offering for collection. That means each transaction must refer to a group of ancestor transactions to deplete their deposit (fees) when verifying the validity. Meanwhile, ancestor transactions must have enough fees for collection. Fees are depleted by incoming transactions from the oldest ancestors in paths to reach a prescribed total amount. High-fee transactions attract powerful miners working in parallel for rapid confirmation, while low-fee transactions will finally be picked up by a small miner. Based on this design, Graphchain makes the current transactions quickly become enshrined in the ancestry of all future transactions.

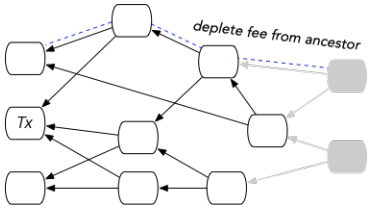

Figure 3: Graphchain

Avalanche [130][182] is a permissionless system based on a new type of approach to reach the consensus. Deviating from BFT-style and Nakamoto mechanisms, Avalanche constructs its underlying

[3]It should be noted that, at the current stage, IOTA relies on a central coordinator [181] issuing milestones to periodically confirm the transactions for stability. They claim the coordinator will be removed in a short future time.

protocol called *Slush*, a CFT-tolerant mechanism, by capturing the concepts from the gossip algorithm and epidemic networks. More specifically, Avalanche randomly samples a small group of nodes to obtains their bias in a bivalent state. Regardless of the size of the network, the size of the sampled group ranges in a small interval (such as from 10 to 20) to facilitate the execution time at reach round. Then, the system arguments Slush to the extended algorithms with a series of parameters (like the single counter in Snowflake and the confidence in Snowball) for finality and security. These parameters assist to yield a threshold result for their historical colors via countering the number of *queries* (as votes in BFT protocols). Last, the augmented algorithm is applied to the whole network which is naturally formed in topological DAG. Thus, the final state, represented by a color in *blue* (b) or *red* (r), is achieved by repeated sampling of the network and customized guidelines of bias. Avalanche also provides a demo to show the process in visualization [164].

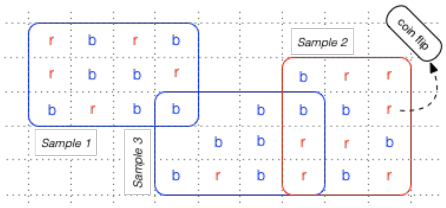

Figure 4: Avalanche

Consensus on Type II. Type II systems are based on blocks. Blocks are structured a natural expanding graph. This part includes systems of Spectre [24], Phantom [80], and Meshcash [81].

Spectre [24][128] is a permissionless network. The key technique behind Spectre is a recursive weighted-voting algorithm based on the precedence of blocks in the underlying topological sort. The voting procedure is completed by blocks, instead of miners. Newly attached blocks are required to submit votes for every pair of blocks, denoted as (x, y), according to their locations. The vote, $(-1, 0, 1)$, over such a pair of blocks, is inherently a preference ordering of the selected blocks, such as x arrives before y (denoted as $x < y$) and vice versa ($y < x$). Consequently, the final decision of the pairwise ordering is measured in the absolute values of aggregate votes (*a.k.a,* weights). A consistent set of transactions is extracted according to the majority of collected votes. The ordering is not necessarily linearizable. This design relaxes the assumption of the ordering where any two transactions must be agreed upon by all non-corrupt nodes. Instead, SPECTRE decides its ordering only by honest nodes. Furthermore, the block renewal rate in Spectre has to slow down, to ensure the voting procedure has been completed by honest nodes.

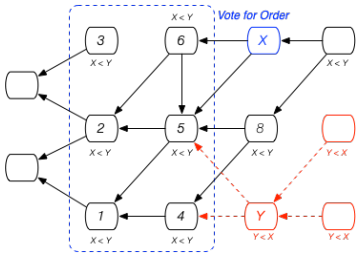

Figure 5: Spectre

Phantom [80] is a PoW-based protocol for the permissionless network. Each block in Phantom contains multiple hash references to predecessors and referenced by multiple successors. The protocol, firstly, identifies a set of well-connected blocks to exclude blocks (with high probability) created by dishonest nodes. Then, Phantom utilizes a recursive k-cluster algorithm (equally, a greedy approximation algorithm) to achieve the partial ordering of the identified set of blocks (cluster) to a full topological order. The greedy algorithm incentives selected blocks inside the cluster while penalizes outside blocks via iterative rounds. The parameter k is used to adjust the level of tolerance of concurrent blocks. After the sorting of blocks, transactions in blocks are ordered according to the order of their appearance. Consistent transactions will be accepted and confirmed as the final state. The iteration over transactions enables a total order of the network. Similar to Spectre, Phantom also relies on honest nodes to agree upon this robust ordering of blocks and transactions. But Phantom differs from Spectre in that it enforces a strict linear ordering over blocks and transactions in the network.

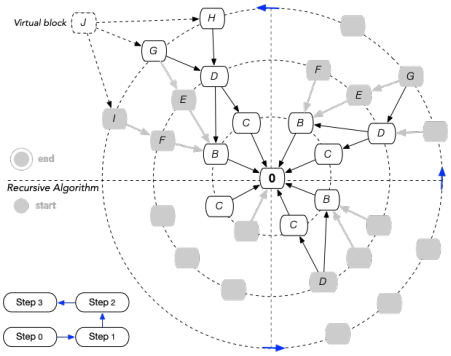

Figure 6: Phantom

Meshcash [81] is a layered DAG system that allows the blocks to coexist at the same time. Firstly, to generate a block, the mining strategy follows classic Nakamoto PoW computations but with the modifications where each block: a) points to every block in the previous *layer* (a specified field as *round* in other systems); b) points to every block with 0 in-degree; and c) increases the counter of layer i when seeing more than threshold blocks in the previous layer. To achieve the consensus, two types of protocols are integrated: a slow PoW-based protocol *tortoise* to guarantee long-term consensus and irreversibility of blocks, and an interchangeable fast consensus protocol *hare* to get quick consensus. A block belonging to which

type of the protocols depends on their appearance in the network, measured by its layer. For blocks in old layers (earlier than $i-s$), the system follows the rules in *tortoise* protocol. The protocol is inherently a leaderless BFT consensus, which decides the consistency (be part of the canonical history) of a block according to the weighted votes of its subsequent blocks. This protocol guarantees a total ordering of blocks. If the same transaction is contained in multiple blocks, the transaction appearing in an earlier block is preferred. For blocks in recent layers (from $i-s$ to latest updates), Meshcash shifts to the *hare* protocol. The protocol is an interchangeable protocol with the aim to quickly settle down the blocks. Meshcash provides both a simple but limited-security method and a complex but attack-resistance one by utilizing the off-chain asynchronous byzantine agreement protocol (ABA). It enlarges the gap of honest and bad blocks and ensures that the honest parties tend to vote in the same direction once completing the protocol.

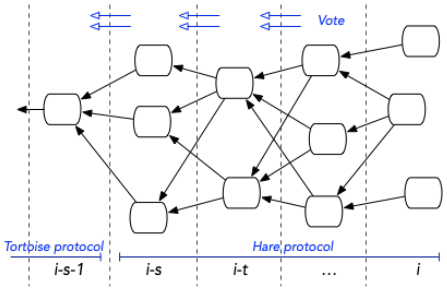

Figure 7: Meshcash

Consensus on Type III. Type III systems are blockless data structure. Transactions are maintained by individual nodes and finally form multiple parallel chains. This part includes systems of Nano [50], Hashgraph [48][183], DLattice [170], Jointgraph [171], Chainweb [147], Aleph [167], Vite [139], Caper [179] and Lachesis-class protocols [122][123][124][125][126].

Nano (RaiBlocks) [50] is a permissionless network with two types of entities involved, namely *account holder* and *representative* (short for *Rpst*). Account holders can select a representative to vote on their behalf in case of offline leaves. Elected representatives serve for solving conflicted transactions. Transactions in Nano are atomic, and an account record all the transaction history related to himself. A complete transfer in Nano consists of two parts – a *send* transaction and a *receive* transaction. First, the sender creates a *send* transaction by referring to the latest block in his account. At this time, corresponding amounts have been deducted from his account. Then, the receiver creates a *receive* transaction by also referring to the *send* transaction and the latest block in her account. When forks occur, a representative creates a vote referencing the conflicted transaction. Then, it starts to observe incoming votes from other representatives. This process lasts for four voting periods and spends one minute in total. Finally, it confirms the winning transaction with the highest cumulative votes (weights, w).

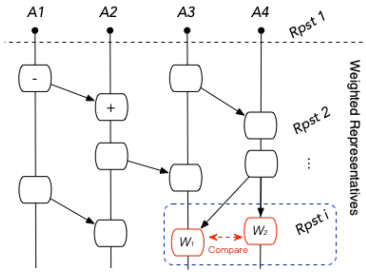

Figure 8: Nano

Hashgraph [48][183] is a permissioned network. Each participated node maintains a separate chain, and nodes mutually interact via the gossip protocol. The node locally creates an *event* to record the history of received information. An event mainly contains three fields, including *timestamp* for synchronization, *transaction* for trading and *hash* for cross-references. Paralleled chains interact with each other based on the design of cross-references. These references point to the latest event on his own chain, and also to the events from synchronized neighbor chains. Events carrying the complete history of ledgers' views are transmitted through the gossip protocol and nodes will eventually obtain a full history. The hashgraph achieves the consensus by a virtual Byzantine agreement consensus. One event to be finally deemed as valid has to go through a three-stage procedure, namely *see*, *strongly see*, and *decide*. Each procedure needs to collect the votes more than a threshold — 2/3 *famous witnesses*, who are the proposers elected by committee members in each round. This design simulates a conventional BFT consensus and integrates the concepts into parallel chains. Events, in this way, can be structured in a global total order. Meanwhile, they are able to reach finality when consensus is completed in current rounds.

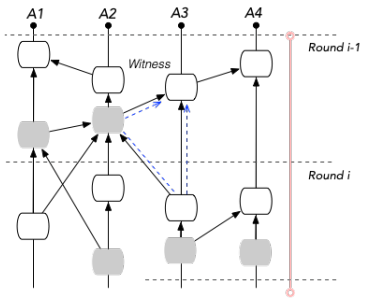

Figure 9: Hashgraph

DLattice [170] follows the structure of Nano and Hashgraph where parallel chains maintained by individual nodes make up a DAG system. The system consists of a Genesis Header and multiple nodes. Genesis Header organizes the involved nodes in the form of Merkle Patricia Tree (MPT), whilst the nodes mainly complete the tasks such as token transferring. Similarly to Nano, the action of a token transferring is split into the *send* transaction and the *receive* transaction between the sender and the receiver. If a node observes any forks of transactions, a consensus program is launched to solve

the conflicts. DLattice uses a so-called *DPoS-BA-DAG* (PANDA) protocol to reach consensus among users. DPOS provides the way of committee formation and BA shows how to achieve the consensus in their DAG. Specifically, the nodes who satisfy the customized PoS condition are able to locally and secretly generate their identities based on the voting power by the Verifiable Random Function (VRF) technique [184]. After broadcasting the messages, other nodes can verify whether an identity is selected as the consensus identity. The consensus is divided into two phases *Vote* and *Commit*. Committee members select a transaction to vote at the Vote phase and start the commit according to collected votes at the Commit phase. If the count of commit votes exceeds the threshold, the consensus reaches on this agreement.

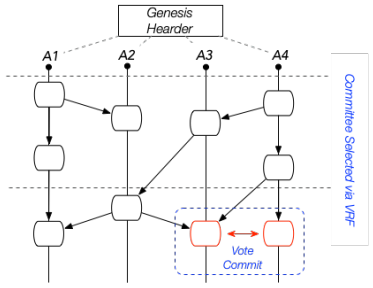

Figure 10: DLedger

Jointgraph [171] is a simplified protocol based on Hashgraph. The consensus in Hashgraph needs at least two voting rounds for events, while Jointgraph, instead, simplifies the voting process into one round by introducing a powerful *supervisor* node. This node mainly monitors the misbehaving nodes by replacing with an honest one, and periodically take snapshots of system states to release the memory. Specifically, each event (carrying transactions) in Jointgraph is broadcast to peer nodes through the gossip protocol. Upon receiving the event from another member, the node will verify its validity (signatures and hash). If all passed, the node votes the event. The finality of this event can be confirmed if it receives more than 2/3 of all the nodes, *one of them must be voted by the supervisory node*. If any conflicts happen, the supervisory node plays the role of a judge to make the final decision. After a certain time of running, two special types of events are issued by the supervisor node, namely the snapshot event and the storage event. These two events are used to take snapshots and store them. Once finishing the permanent storage, the previous memories could be released.

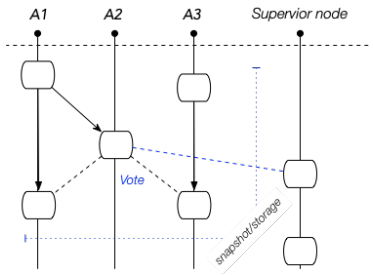

Figure 11: Jointgraph

Chainweb [147] is a permissionless system attempting to scale Nakamoto consensus by maintaining multiple parallel chains. The system has two features. On the one hand, Chainweb uses the cross-reference of hashes to connect parallel chains. Individual chains in the system are based on a PoW consensus that incorporates each Merkle roots from others to increase the hash rate. A block needs to reference the header of its ancestor block, and additionally, reference the headers of peers at the same block height. Thus, each block references all peer chains to form a weaved web according to the base graph (Petersen graph, the network topology of Chainweb). On the other hand, Chainweb employs simple payment verification (SPV) proof to complete token transfers. The procedure follows a similar approach to Nano [50]. Each token to be transferred cross-chain has to move under the SPV proofs. The method forces to destroy tokens from one chain and create equal amounts on another chain. This requires a strict assumption where all chains can grow at a synchronous pace, and fast-growing chains need to be periodically stalled and cleared. As a result, SPV proofs of token transfer can only be validated until the blocks are recorded in other chains.

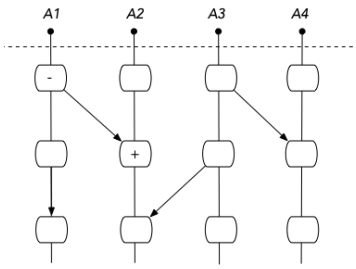

Figure 12: Chainweb

Aleph [167] is a permissioned BFT-style distributed system. The essential idea of Aleph is learned from blockchain systems that adopt the pattern of *Type III*. Specifically, Aleph modifies the classic BFT-consensus by removing the role of the leader. Instead, it enables each node to equally and concurrently issue messages, also denoted as *units*. This design makes the units able to be asynchronously and efficiently transmitted in the network. Similar to the aforementioned systems, The units issued by the same node naturally form into a chain. The units issued by the different nodes are organized in parallel chains and each of them is independent where they can freely create, disseminate, and vote. The key is to build a collectively total ordering among these units. Aleph utilizes leaderless BFT-consensus to achieve the consensus. Firstly, each unit is configured with a round number, and the units with the round number ranging from $0, 1, ..., r-1$ to r are collected in a batch. The system sorts the batches in linearization according to their round numbers. Then, the units within batches break ties by their hashes. Next, with a uniform set of units, the system conducts voting for each unit in the set, to verify whether all nodes can see this unit. If collecting more than $2f$ votes, the unit is deemed as confirmed and finishes the consensus procedure.

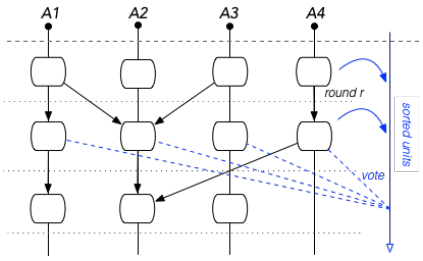

Figure 13: Aleph

Vite [139] follows the basic (lattice) structure of Nano, but introduces a global *snapshot chain* to the achieve the total order sequence. Each account in Vite individually creates transactions in parallel. A complete transaction is split into a trading pair based on an asynchronous *request-response* model, similar to the *send-receive* model in Nano. Transactions can be instantly written into the ledgers without being blocked by the procedure of confirmation. The confirmation is completed by a *snapshot chain*, a storage structure to maintain consistency. Snapshot blocks play the role to store the states (balance, Merkle root, the hash of the last block) of Vite ledgers. The snapshot chain adopts the Nakamoto consensus to structure the blocks in linear. The confidence of confirmation is measured by an accumulated value. The value will increase as the chain grows. If forks happen, the *snapshot chain* breaks the tie by selecting the *longest-chain*. Thus, the transaction is deemed as valid in a probabilistic way.

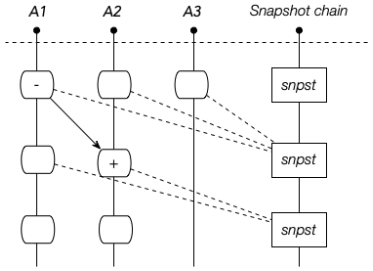

Figure 14: Vite

Caper [179] is a permissoned system with aims to support distributed applications. The system considers both the confidentiality of internal states generated in each application and the interoperability of external states that come from the cross-application transactions. Ledgers in Caper are formed as parallel chains. Deviating from the aforementioned systems, the units are created by different types of applications, rather than nodes. Each application maintains its view of the ledger, including both the internal state and all external states. To achieve the consensus, Caper introduces three consensus protocols by *a separate set of orderers*, *hierarchical consensus*, and *one-level consensus*. These protocols are commonly based on BFT-style consensus mechanisms (CFT, BFT). The mechanism is pluggable and each application can adopt any fitted one according to the specific scenarios. Here, we take the most complex one, hierarchical consensus, as the example. The hierarchical consensus contains two main procedures: a) achieving the local view consensus in each application, and b) reach a global view consensus across multiple applications. Every application operates its local consensus to internally decide on a state. These state from multiple applications are regarded as the input of the global consensus protocol. The global consensus, then, employs an asynchronous Byzantine fault-tolerant protocol to reach the agreement, where all states are structured in a total linear ordered sequence.

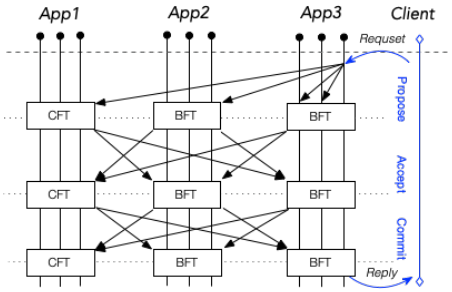

Figure 15: Caper

Lachesis-class [122], denoted by $\mathcal{L}$, is a new family of consensus protocols, designed for distributed networks with features of being asynchronous, leaderless, and resistant against Byzantine fault. This family of protocols integrates a series of algorithms such as peer selection, synchronization, block creation, and consensus. We focus on how to achieve consensus in Lachesis. Generally, the Lachesis has two main steps, corresponded to its hybrid structure. This structure includes two layers. The first layer maintains the DAG structure (called the *Opera chain*) to enable the parallel processing, while the second layer keeps the *Mainchain* to achieve a total order of events. The bridge between these two layers is to select a set of key events, called *Atropos*. A general algorithm of the Opera chain takes multiple events (in the DAG network) as inputs and outputs a selected set of Atropos events. Then, the algorithm of the Mainchain inputs these events and outputs a topological ordering of events. Each node in the network will share this sequence. Thus, all nodes have the same view of the network and the protocol completes the consensus. Lachesis instantiates a family of protocols, including *Fantom* [123], *Onlay* [124], *StakeDAG* [125], and *StairDAG* [126]. The difference between these frameworks is their customized ways to implement the sub-algorithms. More details are shown as follows.

- *Fantom* [123] adopts the leaderless BFT mechanisms to select the key set. In the Opera chain, firstly, several events become the *root* events once collecting more than the threshold (2/3) references of previous roots. Next, a root event becomes the *Atropos* event if more than the threshold (2/3) participated nodes agree on this root event. Last, these Atropos events form the *Mainchain*, a core subgraph of the Opera chain with aims to order the events in the graph. Each Atropos events are created by adding several parameters as their identities including *frame*, *consensus time* and *Lamport timestamp*. In the *Mainchain*, the sorting algorithm is determined based on the allocated identities of each Atropos events, and it follows the priorities of a) the frame value of each Atropos; b) the consensus time when multiple Atropos events coexist at the

same frame; (c) the Lamport timestamp when more than one Atropos having any of the same consensus time on the same frame; (d) the hash value when all previous three conditions stay same.

- *Onlay* [124] employs an online layering algorithm to achieve leaderless BFT mechanisms. This layering provides a better structure of DAG to select the key set. In Onlay, the protocol introduces the H-OPERA chain, built on top of the OPERA chain. To compute the H-OPERA chain, the protocol applies a layering algorithm, such as LongestPathLayer (LPL), to the OPERA chain. The algorithm is inherently a list scheduling algorithm that produces a hierarchical graph with the smallest possible height. A group of Atropos events is, as a consequence, selected to be the key set. As for the *Mainchain*, the protocol orders the Atropos vertices based on priorities of a) their layer, b) Lamport timestamp, and c) hash information of the event blocks.
- *StakeDAG* [125] adopts the PoS mechanism to select the key set, instead of BFT-style protocols. Three steps are included in the Opera chain: a) initialize accounts with stakes, b) computing validation score of an event, and c) assigning weights to new roots. Correspondingly, confirming whether some block is a root in StakeDag is different from that in Fantom: StakeDag requires more than 2/3 of the validating power (total stakes) while Fantom requires more than 2/3 of the total number of nodes. The key set is obtained by such a procedure. The following sorting algorithm in the *Mainchain* is similar to which in Onlay.
- *StairDAG* [126] builds on top of StakeDAG with the difference to distinguish participants into *validators* (high stake) and *users* (low stake) by their stake. Validators can expose more validating power to complete the onchain validation for faster consensus, while users can participant in the system as observers or monitors to retrieve DAG for post-validation. Both of them create and validate event blocks and maintain a DAG network. Similar to calculate the stakes in StakeDAG, each event in StairDAG is associated with a validation score, which is measured by the total weights of the roots reachable from it. The key set is obtained when events have been validated by more than two-thirds of total validating power in the OPERA chain. After that, the sorting algorithm is launched in the same way as StakeDAG.

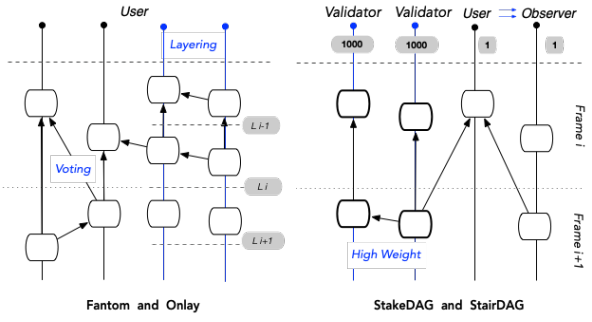

Figure 16: Lachesis-class Protocols

Consensus on Type IV. Type IV systems are block-based structure. Blocks are maintained by individual nodes. This part includes Prism [131], OHIE [145], Blockmania [132], Blockclique [180], Eunomia [146], Dexon [134], and PARSEC [149].

Prism [131] is designed in a similar parallel chain approach. Prism decouples the functionalities of Nakamoto consensus into transaction proposing, validation and confirmation, and utilizes three types of blocks to take these functionalities. Blocks are hence divided into *transaction block*, *voter block*, and *proposer block*. Transaction blocks only generate and carry transactions, acting as the fruit in FruitChain [185]. Voter blocks are used to vote for the proposer blocks and specify a leader block according to their heights. Proposer blocks will pack these transaction blocks and extends the chain under the *longest-chain rule*. The leader (winning miner) confirms the integrity and validity of transactions to form the final ledger. These three types are relatively independent but as well as interacted with each other. Miners will simultaneously work on one transaction block, one proposer block, and multiple voter blocks. A block to be allocated with which type of functionalities depends on a random hash value when the block is successfully mined. In Prism, transactions are confirmed before being ordered, and unrelated ones are simultaneously processed. Meanwhile, a leader block will not be stopped by waiting for its voters becoming irreversible since reverting a majority of voter chains in a short time interval spends much more computing power than reverting one voter chain as in Nakamoto consensus. Thus, Prism, armed with its deconstruction approach, separately scales each functionality to their physical limits for maximum optimization.

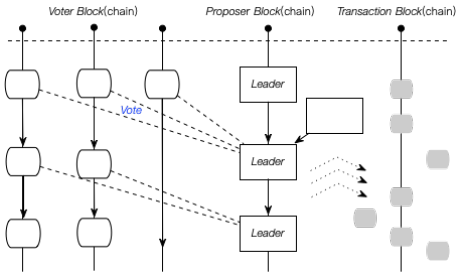

Figure 17: Prism

OHIE [145] shares similarities in composing multiple parallel chains. OHIE is a permissionless blockchain system where chains are equivalent and symmetric. The system operates the standard Nakamoto consensus on each individual chain and allocates each block into these chains according to their hash values. Then, OHIE deterministically sorts the blocks to obtain a global order across parallel chains. To achieve that, the system defines a special tuple embedded in each block, denoted as (Rank,NextRank). The field of Rank represents the current index of a block located on each individual chain, while the field of NextRank specifies a reference to the next block. Here, NextRank is used to balance the length of parallel chains in case of a huge gap between them. It is usually pointed to the block with a high Rank. Blocks with ranks higher than the bar are deemed as totally confirmed, otherwise are partially confirmed. In fact, the tuple together with the chain index inherently locates

a block in the network, and their positions assist to sort blocks in linearization. Generated blocks across parallel chains are sorted by Ranks with a tie-breaking of *chain index*. Specifically, the block with a smaller Rank number will be arranged in front of a block with a higher Rank. This is the first priority. Then, if several blocks with the same Rank number, the block holding a smaller *chain index* will take the advance. Thus, a total linear ordering achieves by adjusting the special Rank fields.

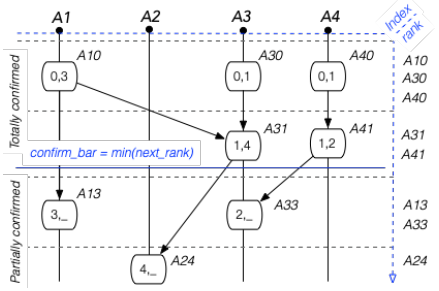

Figure 18: OHIE

Blockmania [132] is a BFT-style protocol which mainly bases on PBFT [186]. Blockmania simplifies the PBFT protocol by a) deciding on a position rather than a sequence number. The position of a block is denoted as $B_{n,k}$, where n is the index of nodes and k is the height of blocks emitted by the node; b) becoming leaderless which is egalitarian for involved nodes; and c) removing several complex procedures like "checkpointing". Nodes in Blockmania separately create blocks by themselves and jointly form the DAG network. Here, each node can only propose one block for the given position without any equivocation. Then, the modified PBFT consensus is executed by these nodes in parallel and a decision is required to execute a classic three-stage (*pre-prepare*, *prepare*, *commit*). Besides, an incentive mechanism, as another essential factor, is introduced to sort transactions in the total order. To solve potential conflicts, each transaction included in a block will be associated with both the height of the block k and deposited fee ϕ. Clients who expect a priority of confirmation can send transactions in earlier rounds or spend higher fees. Upon reaching a decision from sufficient blocks (2/3 threshold), transactions are sorted across parallel nodes according to the tuples $(k, -\phi)$. Alternatively, fees-based tiebreakers may also be replaced by a traditional way — by the value of their hashes, as used in Hashcash [187]. Note that, the formed DAG system is interpreted as a state machine that contains the blocks. Each state machine carrying the interpreted information (view number, a count for prepare and commit messages, stored in square blocks in Fig.19) will be sent and received through blocks. In real network communication, only the blocks are broadcast to peers.

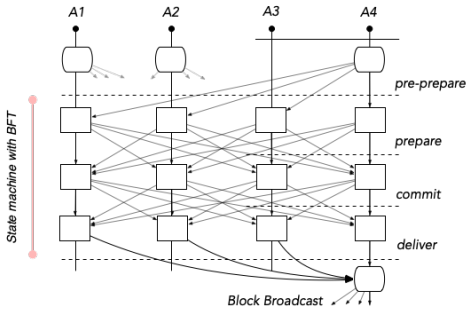

Figure 19: Blockmania

Blockclique [180] shares a similar design of the parallel-chain model, though it names with *multi-thread*. The system is a permissoned protocol which adopts the *proof of ownership of a resource* (PoR, including PoS, PoW, *etc.*) to select its committee members. The selected members take responsibility for creating blocks. Then, Blockclique allocates the received transactions separately into different chains according to their hashes. The transaction sharding avoids incompatibility when blocks are produced in parallel. A block in Blockclique can only pack the transactions with input addresses assigned to this thread, whereas the outputs can point to any thread. Thus, the links between blocks make different threads connected. Meanwhile, Blockclique embeds a scalar *fitness* value for each block to measure the required resources. The fitness is calculated by the total number of involved addresses during the block creation. To achieve the consensus, Blockclique identify two types of incompatibility cases, namely *transaction incompatibility* and *grandparent incompatibility*. Transaction incompatibility represents the conflict transactions that reference the same parent in one thread, while *grandparent incompatibility* means the grandparent references of two blocks are reversed across different threads. Then, avoiding incompatible cases, the system recursively searches for the maximal clique of compatible blocks according to total fitness values contained in blocks. If two cliques hold the same fitness values, the clique with the smallest sum of block hashes is preferred.

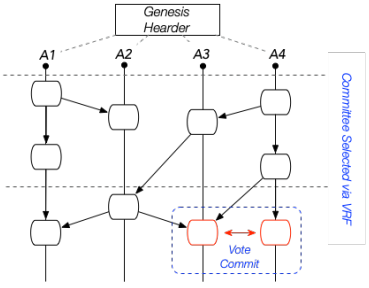

Figure 20: Blockclique

Eunomia [146] is a permissionless system with multiple parallel chains. The system achieves the consensus by the following steps. Firstly, to avoid the conflicts of transactions, Eunomia utilizes a fine-grained UTXO sharding to avoid the double-spending. A transaction can only take the UTXOs with the same sharding index as input

and generate the output UTXOs with random indexes ranging from 0 to $m-1$. Then, to package these cross-reference transactions, miners adopt the *m-for-1* PoW mechanism [178][83] where they can simultaneously create blocks on m chains, but a block only extends one chain for each time. Generated blocks across m chains are sorted by their logical clocks with a tie-breaking of chain index. Specifically, each chain in Eunomia is identified by the chain index i, and the blocks in individual chains are measured in a *virtual logical clock* v, as a counter for increments. Since each chain maintains a local state of the clock, nodes need to involve a hash reference of the updated blocks to synchronize clocks across different chains. The block with the largest v can be recognized in priority. Here, v indicates the epochs to provide separated time slots as mentioned in other protocols [141][148]. Thus, the ordering algorithm based on such positions (i, v) sorts the blocks in the priorities of a) the virtual logical clock v and b) the chain index i.

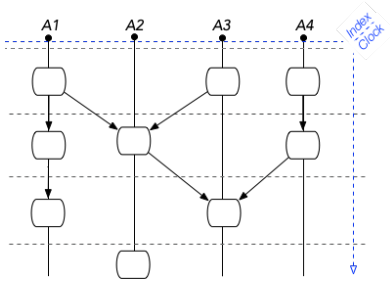

Figure 21: Eunomia

Dexon [134] is a permissioned system. The system is maintained by two special sets of nodes: the *CRS set* to generate public randomness and the *notary set* used to propose blocks. Dexon confirms each parallel chains through the reference field called ack, and determine the global order through these references. Specifically, the consensus of Dexon consists of four steps: a) To avoid the waste of space utility of blocks, Dexon makes each transaction only be packed into one single chain according to the residue of its hash value. This design acts as a *load balance* to schedule the incoming transactions to the individual chains. b) Upon receiving transactions, every single chain individually generates blocks and achieve consensus by employing the technique of a modified Algorand [184] (mainly VRF) scheme. Nodes in CRS set periodically generate updated common random numbers to maintain the variations in each epoch. c) The involved chains form the blocklattice, and each node executes the *total ordering algorithm* with this blocklattice as input. The output is a globally-ordered compaction chain with all blocks sorted in linearization. d) The compaction chain applies the *timestamping algorithm* to compute the consensus time for each block. Then, the consensus timestamp, the height of blocks, and the threshold signature together determine a unique block in the compaction chain. Thus, the validity of the block can be further verified by the following blocks.

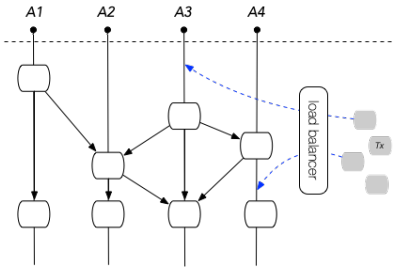

Figure 22: Dexon

PARSEC [149] is a permissioned protocol with the creation of an asynchronous BFT mechanism that is resilient to 1/3 byzantine nodes. The protocol follows the main concepts from Hashgraph [48], where involved nodes spread their events via the gossip algorithm and the consensus of these events is completed by the virtual voting. PARSEC deviates from Hashgraph in that PARSEC additionally defines a new type of data structure called *stable block*. The block inputs a selected set of events after the virtual voting in the graph and sorts them in linearization. Then, it finds the next valid block according to the votes from events. The stable blocks record an order of events that all nodes agree upon, which can also be regarded as periodical checkpoints to confirm the states of the network. The consensus of PARSEC requires two steps: *obtain a set of valid event* from its graph network and *sort the selected events and find the next block*. For the first step, we narrow the focus down to its graph network. In PARSEC, the nodes execute the virtual voting algorithm to decide the qualification of events. The algorithm has two similar procedures as Hashgraph, namely *see* and *strongly see*, to collect the votes (in the form of binary values). The event that receives more than the supermajority (2/3 of total nodes) are considered to be qualified in the next step. Here, the virtual voting elects a set of events agreed by all the nodes as the temporary agreement. Next, for the second step, we focus on the operation surrounding blocks. The key point is to achieve the consensus on both selected events and the next block without any conflicts. The algorithm considers the following three aspects: a) multiple events in the same individual chain are sorted according to their appearance, measured by the embedded counter; b) conflicting blocks seen by multiple events are elected by their collected votes; and c) the tiebreaker is based on the lexicographical sequence.

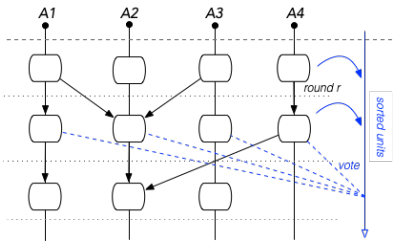

Figure 23: PARSEC

Consensus on Type V. Type V systems are blockless. Appended transactions gradually converge into a main chain. This part includes the systems of Byteball [23], Haootia [82] and *JHdag* [29].

Byteball [23] is a permissionless network. Byteball organizes units (transactions) in the graph topology but later forms a main-chain under the help of trustful and reputable witness nodes. These nodes distinguish common nodes by periodically generating witness units. Each unit is marked with an index called Main Chain Index (MCI) that links to a witnessed unit. Conflicting states are resolved by the main chain index (MCI), where the unit with lower MCI is deemed valid and the higher one fails. If both nonserials hold the same MCI, a tiebreaker rule is applied that the unit with the lower hash value (as represented in base64 encoding) is deemed valid. In Byteball, a total of 12 witnesses are selected to protect against the occasional failures. Witnesses can be replaced by common nodes, who change with better candidates in his list. But the changes happen only gradually since the majority of users are required to achieve an agreement on a new candidate.

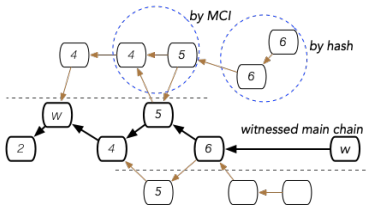

Figure 24: Byteball

Haootia [82] is a permissioned network that consists of three types of roles, *transaction proposers*, *miners*, and *committee members*. Transaction proposers can only send/synchronize transactions; miners can dynamically become the committee members by solving the hash puzzles; committee members are able to participant in its (intra-committee) PBFT consensus process to decide the blocks. In Haootia, a two-layer framework of consensus is induced. The first layer embraces generated transactions and organizes them in a naive graph topology. The second layer is a PoW-based backbone chain with key blocks to decide the total order of transactions. PBFT consensus runs on the backbone chains to achieve the linearization by directly sorting the key blocks, where key blocks confirm the trees of attached transactions (denoted as *increment tree*). Each transaction in the tree may reference multiple ancestors. Only one reference with the smallest lexicographic order could be saved and other parental references are erased. Then, through concatenating the reversed breath-first traverse (RBF-traverse) sequence of the increment trees of key nodes, Haootia achieves an append-only total ordering of nodes.

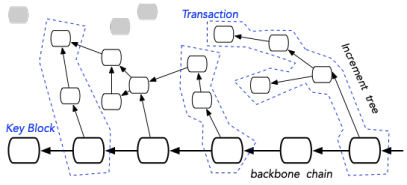

Figure 25: Haootia

JHdag [29] is a permissionless system that applies Nakamoto consensus to the DAG-based protocol. Instead of packing a batch of transactions, each block only contains one transaction to limit the block size and simplify the cryptographic puzzles of PoW. Fast confirmation of blocks is achieved without having to wait for peers (so that we categorize this system in 1^{od}). In JHdag, two types of blocks exist: *common block* for carrying transactions and *milestone block* for making decisions. Becoming which types of blocks depends on their hashes of PoW, where a common block needs 10 consecutive bits of 0 and a milestone block requires 15 consecutive bits. Then, to create a block, each miner specifies three pointers: a) points to the miner's previous block form a peer chain representing the state of that miner; b) points to the previous milestone to form the main chain under the longest chain rule, and c) points to another miner's common block to enhance the connectivity for peer chains. Each milestone blocks can verify multiple common blocks surrounding themselves, and such a clique is denoted as a *level set*. The level set essentially acts as the role of a (Nakamoto) block in Bitcoin, since all blocks in the system are directly or indirectly confirmed by the longest milestone chain. Therefore, the consensus of JHdag inherits similar principles and properties as Bitcoin.

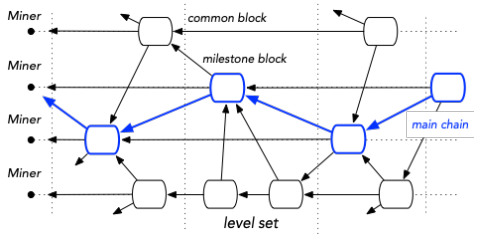

Figure 26: JHdag

Consensus on Type VI. Type VI systems are based on blocks. These blocks gradually converge into the main chain. This part includes protocols and systems of GHOST [45], Inclusive [46], Conflux [141], CDAG [148] and StreamNet [136].

GHOST, Greedy Heaviest Observed Sub-Tree [21][45], is a permissionless network with two types of entities, *common node* and *miner*. Each common node in the network can compete for the rights of packaging blocks, and the winner becomes miner by successfully resolving a hash puzzle as Bitcoin does. GHOST introduces the *weight* to measure the number of blocks in the subtree attached at each block. The difference focuses on the extension rule: the chain grows by recursively selecting the heaviest (weight) sub-tree, instead of the longest-chain. Here, a sub-tree is formed by blocks rooted at the same ancestors. In this scenario, DAG topology is converged to one main chain among several growing sub-trees.

Inclusive protocol [46] is an variant of GHOST, which allows one block to reference multiple ancestors. Only one of the referenced blocks with the same height can be elected as the *father*, earning most of the rewards, while others are denoted as *uncle*, sharing the rest of the rewards. The ancestors are considered within the order of seven generations of the chain. This design makes off-chain transactions involved in the protocol and incentive miners continuously contribute to the network. Ethereum (Casper) adopts this protocol in its implementation.

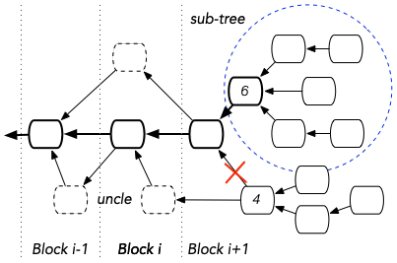

Figure 27: GHOST and Inclusive Protocols

Conflux [51][141] is a permissionless system attempting to scale NC-based blockchain for fast confirmation and high throughput. The system achieves the goal by proposing its improved protocol of GHOST, named *GHAST* (Greedy Heaviest Adaptive Sub-Tree, [141]). The design introduces an adaptive weight mechanism, which enables Conflux to switch between two strategies: an optimistic strategy (similar to GHOST) for fast confirmation and a conservative strategy to resist the liveness attack. Specifically, each block in Conflux has one *parent* edge to form a pivot chain and multiple *reference* edges to operate concurrent blocks. A newly generated block will be adaptively assigned with weights according to its past sub-graph (*a.k.a*, Tree-Graph). If all ancestors in the pivot chain are secured (with a low probability of being reversed), the system assigns 1 to the new block; otherwise, the system assigns a block with weight h within $1/h$ chance (h is a protocol parameter configured as 600 [141]). Blocks with their weights are organized in several sub-trees, and the pivot chain grows by selecting the heaviest sub-tree among them. Every block on the pivot chain is marked with one epoch. Then, Conflux derives a total order of blocks. The priority of sorting blocks follows the metrics: a) their epochs, b) topological order, and c) PoW quality or block hash. Last, the priority of sorting transactions follows: a) the orders of enclosed blocks, and b) the time of the appearance in the same block.

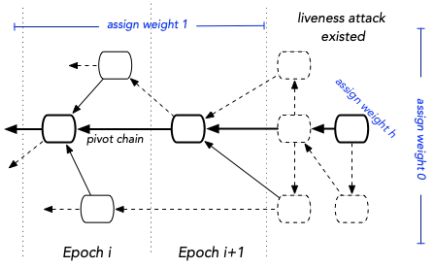

Figure 28: Conflux with its consensus GHAST

CDAG (Converging Directed Acyclic Graph) [148] is a permissioned network consisting of two types of entities, namely *Colosseum* and *block proposer*. Colosseum is designed for permissioned blockchains where the identities of participants are known. Members of Colosseum compete to become the block proposer in a fair and randomized two-player game for each time slot (contain $log_2 N$ where N is the number of participants asynchronous rounds). These proposers partition the transactions into non-intersecting buckets and then select a random bucket to generate blocks and disseminate them into the network. The generated blocks are required to be consistent with a *Converging Block* (C-Block), which is a collection of blocks proposed in a time slot and works as a single point of reference for the next set of blocks. C-Block in each slot enables the system to progress as a chain and maintains a total ordering among blocks. Temporary forks that may frequently occur in the formed chain are solved by using a variant of the heaviest chain selection approach [45].

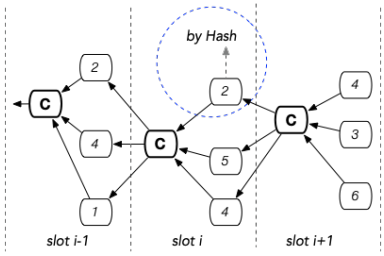

Figure 29: CDAG

StreamNet [136] adopts the concept from Conflux [51] and IOTA [22] to form a permissionless network with a main chain. The system combines the rule of tip selection from IOTA (two ancestors are approved by one transaction) and the structure from Conflux (a pivot chain is required to sort blocks). Specifically, StreamNet consists of multiple *machines* where each machine locally creates the block and instantly broadcasts the blocks through a gossip protocol. To achieve the consensus, a new attached block will approve two ancestor blocks: one is deterministic, and the other is random. The first selected ancestor is deemed as the *parent* block to construct a pivot chain for the purpose of total ordering. The algorithm recursively advances to the previous child block with the highest *score* child of sub-tree. In case of the same score between two child blocks, the one who has the largest hash value wins. The second ancestor is randomly selected through the technique of Markov Chain Monte Carlo (MCMC) to scale out. The random walk starts from the genesis to the lasted attached blocks. Next, a total ordering algorithm is conducted recursively to order the blocks in previous epochs. The algorithm breaks ties by comparing the hash of each block. Upon every machine reach a unified view of DAG, the globally total ordering is completed.

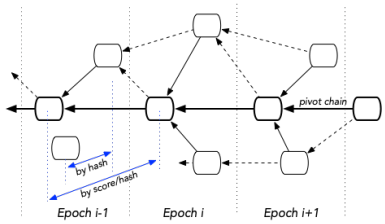

Figure 30: StreamNet

4.3 Summary

Table.3 provides the summary of aforementioned systems. This subsection provides featured techniques and insights. We abstract featured techniques in each system, and some of them are frequently adopted. Details are shown as follows.

Cross-referencing enables chains "twisted" together for improved throughput, high scalability, or low confirmation times. This technique enables the systems to involve orphaned units and increase the in-degree and out-degree of a single unit. As shown in the Table.3, we use (x, y) to show that each unit can reference multiple ancestors, as well as be referenced by multiple successors, where x and y are flexible positive integers larger than 1. Several systems explicitly define the rules of in/out-degree, such as $(x, 2)$ in IOTA, Hahsgraph [48] and StreamNet [136]. Furthermore, the references might be allocated with different roles, which generally includes the *parent* edge and *reference* edges. This design is adopted by Conflux [141] and StreamNet [136] to form the main chain, or by Chainweb [147] and OHIE [145] to form parallel chains.

Trusted authority represents the powerful authority who makes the final decision. The authority can be instantiated as the *leader* in PBFT [186], the *coordinator* in IOTA [22], the *snapshot chain* in Vite [139], the *supervisory nodes* in Jointgraph [171], the *representatives* in Nano [50], the *proposer block* in Prism [131], and the *witness nodes* in Byteball [23]. Trusted authorities can either directly affect the consensus or indirectly solve the conflicts. This technique speeds up the confirmation of units but sacrifices the decentralization, bringing additional risks. The adversary may launch several types of attacks against these centralized parties.

Pairwise vote is a 2-for-1 voting selection, rather than n-for-1 in normal voting algorithms. Each unit according to its knowledge decides one element has a high priority than another. The procedure only guarantees partial consistency. This technique is adopted by Spectre [24] as the consensus, as well as by several systems to solve the conflicts when sorting the units. Additionally, the pairwise vote differs from the *coin-flip* [75] in their randomness. Pairwise vote relies on the willingness and knowledge of the node, whereas coin-flip is based on a mathematically random selection with even distribution between two values.

Transaction sharding is to allocate the transactions into different chains, in advance, to prevent potential duplication and conflict in sorting procedures. The technique is adopted by the models of parallel chains, including Blockclique [180], Eunomia [146], and Dexon [134]. In general cases, the allocation follows the rules according to the residue of the transaction hash. Transaction sharding relies on an implicit assumption that each transaction will be only processed by one miner/validator in the network.

The PoW mechanism in some DAG-based systems such as IOTA [22], Nano [50], Hashgraph [48], is simply used as an anti-spam tool which can be computed within seconds. PoW for subsequent blocks is pre-computed once the transaction is sent. This design enables transactions instantaneous to an end-user.

Insights. We further discuss the consensus with the following types. This provides a general perspective when designing and analyzing DAG-based blockchain systems.

- ***Tip selection algorithm*** is a set of rules specifying how an ancestor transaction selects its subsequent transactions, or how a new transaction attaches to the ancestors. The algorithm is widely adopted by the systems in topology $\widehat{D}$, such as IOTA [22], Graphchain [79], and StreamNet [136]. The advantage of applying TSA in DAG systems is to replace complex consensus mechanisms with the simplest rules. This design enables higher throughput and better scalability, and tolerable resistance against forks. But as sacrifices, they reduce security and strict consistency to some degree.
- ***Recursive algorithm*** represents a family of algorithms that makes each round's outputs progressively approach to stable values. Recursion obtains the result by using functions that call themselves several times. This design is widely adopted by consensus mechanisms to enable the disordered units convergence into a total sequenced chain. The systems can extend the graph in a determined way. This family of algorithms include the *recursive transverse algorithm* in Spectre [24]/Blockclique [180], *greedy algorithm* in Phantom [80], and *sampling* algorithm in Avalanche [130].
- ***BFT-style consensus*** in DAG systems can be classified into three types. The first is to directly apply the classic BFT consensus protocols (PBFT, BFT, CFT) into the system. It requires selecting a committee in advance to operate the consensus. The committee members are selected according to their owned resources (PoW, PoS). The approach is adopted by the systems of Blockmania [132], Dexon [134], and Haootia [82]. The second type is modifying the protocols to fit for parallel chains. It is acknowledged as *async-BA*, and sometimes known as *leaderless BFT* protocols. The key design of this type is to remove the single-leader related phrases from the protocols. Each chain can freely produce and broadcast blocks and vote for anyone he favors. Any block that collects enough (2/3 of total) votes is deemed as confirmed. Since the blocks in each chain get confirmed and confirm peers in asynchronization, it is difficult to achieve a global linear order. Additional techniques are required, such as the *gossip by gossip* in Hashgraph [48], or the *sorting algorithm* in PARSEC [149], Fantom [123]. The third type is integrating the former two types to form a *hierarchical* protocol such as Caper [179]. Classic BFT protocols are first applied to each separated zone, and the async-BA is adopted by an upper layer protocol to achieve the final consensus across zones.
- ***Nakamoto consensus and its variants*** are the most prevailing consensus approaches in current blockchain systems. Classic NC selects the longest sub-chain whilst the variant NC selects the highest weighted sub-chain. NC and variant-NC are adopted by the systems that can form a pivot (main) chain, such as GHOST [45], Inclusive [46], Conflux [141], Vite [139], CDAG [148], and Prism [131]. This design relies on complex rules to extend the chains and then sort the rest of the units in each round. NC/variant-NC can also be applied to multiple chains that simultaneously grow, such as Eunomia [146], JHdag [29], and OHIE [145]. But singly NC/variant-NC, in this case, cannot guarantee the linear order of units so that additional techniques are necessary.
- ***Sorting algorithm*** aims to sort the units in total linear order. This is essential to guarantee global consistency. Sorting

algorithm relies on several types parameters, including cumulative confidence like *weight, score, fee*, random beacon like *hash*, or natural sequence like *lexicographic order* and *appearance*. Then, the algorithm sorts the units by setting a priority among these parameters. For example, Conflux [141] organizes the blocks under the priorities of a) their epochs, b) topological order, and c) PoW quality or block hash. This procedure is used as a complementary mechanism to sort filtered units after NC or BFT protocols. Half of DAG systems apply the sorting algorithm to their protocols for strict consistency and better security. A linear order of units would support upper layer constructions such as the state-transited smart contract.

5 PROPERTY ANALYSIS

Properties reflect the inherent protocol designs. We capture three properties related to the consensus procedure, namely *consistency*, *ordering* and *finality*. Other properties (that are potentially related, but not frequently mentioned) are omitted in this paper. In the following subsections, we respectively provide the reasons, definitions, and analyses of these properties.

5.1 Properties Between BA and NC.

Byzantine Agreement (BA-style) [186][188] and Nakamoto consensus (NC) [1] are the most prevailing protocols adopted by current blockchain systems [16][176]. BA-style protocols have been proposed to achieve consensus in the presence of malicious nodes, where the tolerance is maximal 2/3 of the total nodes [186]. NC protocols, in recent decades, stand out in one critical aspect thanks to its remarkable simplicity. NC introduces a new method to extend the chain – the longest subchain wins. This design breaks the assumption in BA-style protocols that only the closed committee can conduct the consensus. Instead, it enables all participants to get involved in the consensus process by the ways like PoW [1], PoS [189], *etc* (PoX). These two types of protocols are radically different in the mechanism designs, but they follow similar rules in properties. Here, we start our observation.

Classic BA-style protocols follow three well-known properties: ($a1$) *agreement*: for a given consensus instance, if a correct node decides the block B, then all correct nodes decide B; ($a2$) *validity*: if all correct nodes propose a valid transaction before starting a consensus instance, then the block decided in this instance is not empty; and ($a3$) *termination*: all correct nodes eventually have the decisions. Nakamoto consensus slightly modifies the definitions. Two key properties of NC protocols [178][10] are: ($b1$) *persistence*: if a transaction is seen in a block deep enough in the chain, it will stay in the position; and ($b2$) *liveness*: if a transaction is given as input to all honest nodes, it will be eventually inserted in a block that is deep enough in the chain.

Properties between ($a1, a2, a3$) and ($b1, b2$) are different. The difference roots in their system formulations. The BA-style protocols rely on the interactive model of communication, where transmitted messages need to be responded from peers. The decision, which cannot change anymore, is completed once enough participants agree on it. In contrast, Nakamoto consensus is based on the non-interactive model and the states get updated in every phase. The probability of being reversed depends on its depth. The formulation of NC protocols shifts from the send-receive replication to state machine replication [178][190]. These modifications make the above-mentioned properties presented in different forms.

However, in fact, the properties from BA and NC essentially pursue the same goals. Firstly, the *liveness* property of NC ($b2$) is phrased as chain growth and chain quality, and a block is inserted in the chain with a negligible probability of being reversed. As one block is sufficiently buried in the chain, the property $a3$ and $b2$ could be regarded as approximately identical. Several studies even provide the boundary of reversibility [190][191]. Secondly, the *consistency* property ($a1$ and $b1$) ensures that a system can obtain the global view of states. All blocks see the same blocks at a specific height. Decisions in BA-style consensus are deterministic since peers are allowed to negotiate at each round. Decisions in NC protocols are probabilistic. The confidence increases as the chain develop under the *longest chain wins* rule. Thirdly, the *validity* property ($a2$) is guaranteed by the verification of hashes in each block header, as a default setting in systems.

Their similarities enable us to inherit the common principles of both BA-style and NC properties and also left gaps to add specialized properties concerning DAG. Specifically, we provide three properties, namely *consistency*, *ordering* and *finality*. The *consistency* is similar to the properties of $a1$ and $b1$, while the *finality* matches $a3$ and $b2$. The property *ordering* is added as the distinguished property when adapting to DAG-based blockchain systems. This property is neglected in BA-style consensus and NC protocols due to their implicit structure: all units in their systems must be sorted in linearization. Details are shown in the next subsection.

5.2 Properties

Consistency. Scalability and performance in blockchain systems are closely related to the requirements of consistency. Smart contract-supported platforms rely on strict consistency. The state transition must be aligned in a linear sequence since the state-disorder may fail operations. In comparison, some scenarios may only require a weak assumption – partial consistency, where the sortation happens in associated transactions, rather than all transactions. It fits the scenarios that do not provide strict integrity, such as the proof of existence in certificate systems, the token transferring inside organizations. Here, we define two types of consistency as follows.

- ***Strict consistency.*** Each honest node has the same decisions for a specified position[4] of the ledger. For example, BA-style consensus and Nakamoto consensus rely on strict consistency by default.
- ***Partial consistency.*** Only associated nodes[5] have the same decisions, whereas the decisions of disjoint nodes are unknown to each other. The number of associated transactions ranges from two (pairwise order) to multiple.

Comparison with Global View. The view of a ledger/node represents its corresponding local state. The strict consistency does not necessarily require an arbitrary node has a global view since the assumption of *global* is much stronger than that of strict consistency. Assuming that all nodes maintain the same ledger (like

[4] A specified position indicates the units can be precisely positioned in the system.
[5] Associated nodes mutually interact by the means of associated transactions.

Table 3: The Structures, Consensus, Properties of DAG-based Blockchain Systems

	Structure					Consensus							Property		
	Types	*Unit Representation*	*Topology*	*In/Out Degree*	*Transaction Model*	*Openness*	*Membership Selection*	*Unit Allocation*	*Unit Positioning*	*Extension Rule*	*Conflict Solving*	*Featured Technique*	*Consistency*	*Ordering*	*Finality*
IOTA [22]†‡	*I*	Bundle	$\widehat{D}$	(x,2)	UTXO	.less	-	-	-	TSA(MCMC)	Weight	Blockless	◐	◐	◐
IOTA [181]★	*I*	Bundle	$\widehat{D}$	(x,2)	UTXO	.ed	Assign	-	-	TSA(MCMC)	TR	Coordinator	●	◐	●
Graphchain [79]†‡	*I*	Tx	$\widehat{D}$	(x,y)	UTXO	.less	-	-	-	TSA	Fee	Incentive	◐	◐	◐
Avalanche [130]†‡	*I*	Tx	$\widehat{D}$	(x,y)	Acct	.less	-	-	-	Sampling	Query	Coin-flip	◐	◐	●
Spectre [24]†‡	*II*	Block	$\widehat{D}$	(x,y)	Acct	.less	-	-	-	RTA	Vote	Pairwise vote	◐	◐	◐
Phantom [80]†‡	*II*	Block	$\widehat{D}$	(x,y)	Acct	.less	-	-		GA	Score	k-cluster	●	●	◐
Meshcash [81]†	*II*	Block	$\widehat{D}$	(x,y)	Acct	.less	-	-	-	async-BA	-	Layered ptcl	◐	◐	●
Nano [50]†‡	*III*	Tx	$\widehat{P}$	(1,1)	Pair	.less	Elect	-	(i,h)	Natural♮	TRs	Trading pair	◐	◐	◐
Hashgraph [48]‡♭	*III*	Event	$\widehat{P}$	(x,2)	Acct	.ed	Elect	-	(i,h)	async-BA	Witness	Gsp by Gsp	●	◐	●
DLattice [170]‡‡	*III*	Tx	$\widehat{P}$	(1,1)	Pair	.less	DPoS	-	(i,h)	async-BA	-	Trading pair	◐	◐	●
Jointgraph [171]†‡	*III*	Event	$\widehat{P}$	(x,2)	Acct	.ed	Assign	-	(i,h)	async-BA	TR	Supervisory	◐	◐	●
Chainweb [147]†	*III*	Tx	$\widehat{P}$	(x,y)	Acct	.less	PoW	-	(i,h)	Natural	Length	Cross-Ref	◐	◐	◐
Aleph [167]†	*III*	Unit	$\widehat{P}$	(x,y)	Acct	.ed	Elect	-	(i,h)	async-BA	Hash	Sorting ptcl	●	●	●
Vite [139]‡	*III*	Tx	$\widehat{P}$	(1,1)	Pair	.less	Assign	-	(i,h)	NC	TR	Snpst chain	●	●	●
Caper [179]†‡	*III*	Tx	$\widehat{P}$	(x,y)	Acct	.ed	Assign	App§	(i,h)	async-BA	-	Hierarchical	●	●	●
$\mathcal{L}$-Fantom [123]†	*III*	Event	$\widehat{P}$	(x,y)	Acct	.less	Elect	-	(i,h)	async-BA	Apprnce	Sorting ptcl	●	●	●
$\mathcal{L}$-Onlay [124]†	*III*	Event	$\widehat{P}$	(x,y)	Acct	.less	Elect	-	(i,h)	async-BA	Apprnce	Layered DAG	●	●	●
$\mathcal{L}$-StakeDag [125]†	*III*	Event	$\widehat{P}$	(x,y)	Acct	.less	PoS	-	(i,h)	Natural	Weight	Validator	●	●	●
$\mathcal{L}$-StairDag [126]†	*III*	Event	$\widehat{P}$	(x,y)	Acct	.less	DPoS	-	(i,h)	Natural	Weight	w-Validator	●	●	●
Prism [131]†‡	*IV*	Block	$\widehat{P}$	(x,y)	Acct	.less	Elect	Role	(i,h)	NC	TR	Decoupling	●	◐	●
OHIE [145]†‡	*IV*	Block	$\widehat{P}$	(x,y)	Acct	.less	PoW	-	(i,h)	NC	Rank	Sorting ptcl	●	●	◐
Blockmania [132]†‡	*IV*	Block	$\widehat{P}$	(x,y)	Acct	.less	PoS	-	(i,h)	smpl-PBFT	Fee	Positions	◐	◐	●
Blockclique [180]†‡	*IV*	Block	$\widehat{P}$	(x,y)	Acct	.less	PoR	Chain	(i,h)	RTA	Fitness	Tx sharding	◐	◐	◐
Eunomia [146]†	*IV*	Block	$\widehat{P}$	(x,y)	UTXO	.less	PoW	Chain	(i,h)	NC	Clock	m-for-1 PoW	●	●	●
Dexon [134]†	*IV*	Block	$\widehat{P}$	(x,y)	Acct	.ed	Assign	Chain	(i,h)	BA	Apprnce	VRF	●	●	●
PARSEC [149]†	*IV*	Event	$\widehat{P}$	(x,y)	Acct	.ed	Assign	-	(i,h)	async-BA	LO	Stable block	●	●	●
Byteball [23]†‡	*V*	Unit	$\widehat{C}$	(x,y)	UTXO	.less	Elect	-	-	Natural	TRs	Witness node	●	◐	●
Haootia [82]†‡	*V*	Unit	$\widehat{C}$	(x,y)	Acct	.less	PoW	-	(h,i)	PBFT	LO	Hybrid ptcl	●	●	●
JHdag [29]†	*V*	Tx	$\widehat{C}$	(x,y)	Acct	.less	PoW	-	(h,i)	NC	Hash	Flexible PoW	●	●	◐
GHOST [45]†‡	*VI*	Block	$\widehat{C}$	(x,1)	UTXO	.less	PoW	-	(h,i)	variant-NC	Apprnce	Earliest DAG	●	●	◐
Inclusive [46]†‡	*VI*	Block	$\widehat{C}$	(x,y)	UTXO	.less	PoW	-	(h,i)	variant-NC	Apprnce	Involve Uncles	●	●	◐
CDAG [148]†‡	*VI*	Block	$\widehat{C}$	(x,y)	Acct	.less	PoW	-	(h,i)	variant-NC	Hash	C-Block CDAG	●	●	◐
Conflux [140]†‡	*VI*	Block	$\widehat{C}$	(x,y)	UTXO	.less	PoW	-	(h,i)	variant-NC	Apprnce	Adaptive ptcl	●	●	◐
StreamNet [136]†	*VI*	Block	$\widehat{C}$	(x,2)	Acct	.less	-	-	(h,i)	TSA(MCMC)	Weight	Pivot chain	●	●	◐

● = totally provides property (global consistency; linear Sorting; deterministic finality, respectively).
◐ = partially provides property(partial consistency; topological sorting; probabilistic finality, respectively).
− = does not provide property; ★ = Temporary Settings. ♮: This represents the tip attachment procedure without any predefined rules.
† = Academic documents (Preprint included) available; ‡ = Implementation available; ♭ = Patent Protection; default = Whitepaper/Concept.
§: App: Units are identified in different applications; Role: allocate the unit with the role (functionalities); Chain: allocate the unit into an individual chain.
(x,y) represents the number of in-degree and out-degree edges, satisfying $1 < x \leq m, 1 < y \leq m$ where m is the maximum number of generated units at one round.
h = unit index/height/clock in the (sorted) chain; (i,h) = the unit with index/height/clock h in the i-th chain.
Abbreviation: Tx = transaction; Acct = account; ptcl= protocol; strc = structure; smpl = simple; Gsp = Gossip; Apprnce = Appearance; w-Validator = Weighted validator.
TR = Trusted Role (a powerful authority who makes the final decisions, such as leader, coordinator, *etc.*); TRs = a small group of TR; LO = lexicographic order;
TSA = Tip Selection Algorithm; RTA = Recursive Traverse Algorithm; GA = Greedy Algorithm; async-BA = asynchronous Byzantine Agreement mechanisms; variant-NC = derived NC protocols which adopt the weightest subgraph; MCMC = Markvo Chain Monte-Carlo [157]; Adaptive ptcl = Adaptive weighted protocols.

Bitcoin[1]/GHOST[45]/*etc.*), the property of strict consistency can be approximately regarded as a global view because all nodes synchronize the same (latest) states that have been agreed and synchronized to their local chains. In contrast, if every node only maintains its own ledger (like Nano[50]/OHIE[145]/Prism[131]/*etc.* in *Type II-I/IV*), the property of global views can hardly be achieved due to the isolation of their units. But achieving strict consistency is feasible with the help of special operations such as the sorting algorithm.

Ordering. Ordering indicates how the units in the systems are organized. Classic blockchain systems arrange the units in linearization as default. In contrast, DAG-based systems break this implicit assumption by introducing the concept of graph-based structure. Units attached to the network may form any shapes of topologies, either the divergent graph or parallel-line graph. The difference between DAG systems is whether they sort the disordered units into a linear sequence with an additional step. We use the term *topological sorting* to describe the systems that stay the status of disordered units. We define two types of ordering to distinguish the desired property as follows.

- ***Topological ordering.*** The topological ordering in a DAG system is an ordering of the vertices satisfying the properties of *unidirectional* and *cyclic* as defined in Equation 1.
- ***Linear ordering.*** Besides satisfying the properties of *unidirectional* and *cyclic* in topological ordering, linear ordering additionally requires the vertices (units) in the network being ordered in a *total linear sequence*. BA-style consensus and NC protocols rely on the linear sorting by default.

Approaches to achieve Linear ordering. The systems, where multiple nodes maintain the same ledger, sort units recursively. This means the sorting algorithm runs instantly once a unit is attached to the main chain (Phantom[80]/Aleph[167]/GHOST[45]). Every unit will be recognized with a height or an index after the serialization. In contrast, the systems, where each node only maintains its ledger, rely on a periodical way to sort the units. The systems collect the qualified units (*e.g.* with sufficient votes) during each time interval. Then, these units are sorted according to their confidence (appearance time, weight, hash, lexicographical order, *etc.*) into a linear sequence. The sorting algorithms also might be operated by trusted authorities (Vite[139]/Jointgraph[171]). But in most cases, they are embedded in the default systems.

Features of Topological ordering. Topological ordering includes all the typologies except for linearly structured ones. The systems in *Type V/VI* are inherently designed with the main chain for unit linearization. In contrast, topological ordering appears in systems belonging to *Type I-IV*. We have identified two types of topological ordering from the perspective of whether an arbitrary unit can be traced back to the genesis unit or not. Transitive systems includes IOTA [22], Graphchain [79], *etc.* Otherwise, a user can only trace to a certain transaction in the graph. This frequently appears in the pairwise-ordering systems, such as Nano [50], Spectre [24] and DLattice [170]. Topological ordering limits the systems only suited to support cryptocurrency networks where the global state transition is not a necessity.

Relation with Consistency. If the units in a system are organized in linear ordering, this system can achieve strict consistency. The reason is straightforward: every unit in the formed chain can be uniquely identified and recognized. The decision (including the history of previous states) of a specific position is shared by all the nodes without conflicts. However, in topological sorting, we cannot derive similar results because the properties in each type of typology are significantly varied. We only discuss the systems in *convergence* topology. The *convergence* states the idea that the generated units are expected to close to a certain distribution. Units, being organized in linear sequence or tending to be organized in a linear sequence, could be denoted as *convergence* systems. Thus, in our classification, the systems in *Type V/VI* can reach strict consistency since all the units will be eventually organized in linear ordering.

Finality. Consensus finality (*a.k.a.* "forward security" [192]) represents the property where a confirmed unit cannot be removed from the blockchain once successfully appended to his parent unit at some point in time. Formally, as defined in [193], *if a correct node i appends a unit b to its copy of the blockchain before appending the unit b', then no correct node j can append the block b' before b to its copy of the blockchain.* A block will be eventually either fully abandoned or fully adopted. For traditional BA-style consensus, such decisions are instantly made through two/three rounds of negotiation between leader and replicas. In contrast, a block in NC protocols is appended in the chain with the risk of being reversed. The possibility is decreased along with the increased depth that this block being buried. We abstract two types of finality when adapting to the DAG-based systems.

- ***Probabilistic.*** The units appended to the system are always accompanied by the risk of being reversed. The probability is inversely proportional to its cumulative confidence (including the forms of depth/score/weight *etc*). The confidence is obtained from the contributions of subsequent units.
- ***Deterministic.*** The units, once attached to the system at some point in time, will instantly and permanently get confirmed so that they can never be removed from the system.

Comparison with Liveness. Liveness, as we discussed in the previous content, ensures that if a unit is given as input to all honest players, it will eventually be inserted in the chain. The property of liveness focuses on the problem: *whether the systems can continuously run without crash or faults*. Network failure and malicious attacks are the two major threats against liveness. In contrast, finality guarantees that the unit inserted in the chain will be reversed in a negligible probability. This is equal to say the units are buried deep enough. The property of finality emphasizes the problem: *whether attached units are permanently valid in systems.*

Approaches to achieve Deterministic Finality. Achieving instant and deterministic finality relies on trusted roles (TRs) in current DAG-based systems. TRs can be instantiated as the forms of leader, validator, coordinator, *etc.* The way to elect TRs and the principles to be followed distinguish different systems. We give several examples. *Byteball* [23] relies on a set of reputable and honest witnesses to determine a main-chain with finality. Nano [50] provides quicker and deterministic transactions by a group of voted representatives via a balance-weighted vote on conflicting transactions. Vite [139] makes assigned TR create a snapshot chain to record the transactions from common nodes. The finality in Prism [131] is determined by its leading proposal blocks.

6 SECURITY ANALYSIS

This section provides several types of attacks with their assumptions and scopes. We list a few of the leading systems mentioned in the literature as instances to show how to mitigate these attacks. A summary is presented in Table.4.

6.1 Parasite Chain Attack

Parasite chain attack [22] shares a similar mechanism as selfish mining in Bitcoin [194]. The attack attempts to replace an honest subgraph with a prepared subgraph (*a.k.a.*, parasite chain) for more profits. To launch the attack, an adversary secretly generates a subgraph offline but occasionally references the main graph to obtain a high score (equally weights/confidence/votes *etc.*). Then, the adversary sends a pair of conflicting transactions separately to the main graph and his private subgraph. He continues to work for a while ensuring that his subgraph collects competitive scores for games. At this time, the conflicting transaction in the main graph may get confirmed (money spent) by several approvals from honest tips. The adversary at the same time publishes his prepared subgraph to invalidate the main graph and wins the competition with a high probability. Thus, a coin has been spent twice.

This attack relies on the assumption where an adversary has sufficient computing power to efficiently generate the units. Based on that, the attack targets the protocols without instant finality by powerful leaders. We show how IOTA overcomes this attack.

Instances. The tip selection mechanism plays a critical role in the security of IOTA [22]. IOTA proposes a weighted random walk (MCMC) to adjust the transition probability by a parameter α. Theoretically, if α is high enough, the system is secure under the assumption (malicious power $< 51\%$). However, an excessively high α makes the main graph become a chain, with lots of transactions being orphaned. The studies of [114] proposes a modified tip selection mechanism to resist the parasite chain attack. The modification focuses on the explicit formulation of the MCMC algorithm, where they employ the *derivative* of original MCMC equations, called First Order MCMC. They conduct a set of experiments to simulate the effect on the likelihood of an MCMC walk terminating on tips. The derivative of MCMC formulations makes the cumulative weight on the main graph grow at a different rate. When the particle (x-axis) is far away from the tips, the algorithm has significant effects, whereas the particle approaches tips, the algorithm performance starts to deteriorate. With the algorithm, the parasite chain grows linearly with the rate of attackers' computing power, whilst the main graph grows at the rate of rest power of the network. The parasite chain, thus, gets heavily penalized in the modified mechanism since the power of attackers is less than the honest participants. Additionally, a model-based detection mechanism against this attack in IOTA is provided by [165].

Spectre [128] describes a type of attack which refers to the malicious behaviors of the miners. We classify this attack as parasite chain attack in our model due to their similar approaches, although Spectre named it with the censorship attack. In the normal case, miners have to honestly verify recent blocks and immediately publish their blocks. But in this attack, dishonest miners deliberately *ignore* certain blocks and transactions, preventing the blocks from being accepted due to the lack of enough votes. The merchants may consider the transactions contained in such blocks are unsafe. This potentially delays the acceptance rate of transactions or even invalidates the main subgraphs. However, a successful attack requires an instant *Poisson bursts* in block creation to make the blocks generated by attackers outpace others. The honest node can extend the waiting period of accepting transactions to minimize the probability of such bursts. The correct votes will eventually exceed the malicious votes since honest nodes occupy the majority. Each round of iteration strengthens the decision of correct votes that consistent with the majority of past blocks. Spectre provides the experimental results, showing that the acceptance rate may decreases to some degree when compared to situations without attack, but still at a high-speed level.

6.2 Balance attack

Balance attack [195], also know as *liveness attack*, aims to profitably keep several chains/subgraphs growing in the same pace. The attack partitions the network into multiple subgraphs with balanced computational power. An adversary leverages a dynamic strategy to wander across those balanced subgraphs and selects one favorable fork to maximize his/her profit. To launch the attack, an adversary imposes a (time) delay to the subgraphs with equivalent mining power and select one subgraph to add his issued transactions. The miner in this subgraph performs as normal to collect transactions for block production and offer irrevocable benefits (like services, merchants, goods, *etc.*) to the adversary. In the meanwhile, the adversary accepts the benefits while issuing conflicting transaction in other subgraphs with additional computing power, to guarantee that he can arbitrary let the second subgraph outweigh the original one. The miner dynamically maintains the balance between two (or more) subgraphs until the miner learns the fork. Even though, the adversary is still able to invalidate the miner's subgraph and conduct the double-spending.

This attack heavily relies on the assumption of a powerful adversary who: a) can split/partition the network and add the time delay; b) can be temporarily and arbitrarily wander across different subgraphs without being known by honest miners; c) has enough mining power to extend one subgraph outweighing others. Thus, the balance attack targets PoW-based protocols with high block generation rates. These protocols rely on miners for block/transaction production, which is vulnerable to powerful and strategic attackers. DAG-based systems confront similar threats as it does in Bitcoin, especially for protocols that stem from the Bitcoin protocol like GHOST, Conflux, OHIE, Prism, *etc.* We look through how these protocols mitigate attacks.

Instances. Balance attack strategically delays the confirmation process of new blocks, instead of reverse past blocks as it in parasite attack. We denote the block generation rate as λ and the delay from an adversary is time d. As claimed in [145], an adversary with little computation power stalls the normal procedure of consensus when $\lambda d > 1$.[6] To solve the issue, reducing the rate of block generation is a reasonable way, because a higher block generation interval makes fewer blocks attached to the system. GHOST can tolerant the attack to some degree since the interval between two consequent blocks

[6]The equation is obtained by the following assumption: the propagation delay λ is larger than the block generation interval $\frac{1}{\lambda}$, where $d > \frac{1}{\lambda}$.

requires a long time (10 minutes in Bitcoin). The waiting period limits the behavior of adversaries who rely on delay time d to launch the attack. Thus, low mining rates of GHOST [64][155] mitigate the effectiveness of the balance attack.

In contrast, this attack is effective in the early version of Conflux [51]. The reason [145] is due to its high throughput where massive blocks are generated in each round. If the production of one block requires 1 second while the propagation needs around 10 seconds, around 10 blocks are attached to the system at the approximate same time. Such heavy pressure is imposed on subgraphs where an attacker may take advantage of the chaos. In its updated version [141], Conflux proposes its improved consensus GHAST [140] by providing two strategies to resist the balance attack: an optimistic strategy as GHOST and a conservative strategy. The switch of these strategies is inherently based on an adaptive weight mechanism. When detecting a divergence of computing power, GHAST slows down the block generation rate by adaptively switch to the conservative strategy. The gap between these two strategies is that only a small fraction of blocks in the conservative strategy, instead of all, are granted with weights for further branch selection. Other blocks are deemed as valid but set to be zero weight. If we only consider the weighted blocks, the block generation rate is sufficiently low to defend the balance attack. Another critical point is how to detect the balance attack in the network. GHAST relies on its recursive algorithm to see whether its best child has a dominant advantage compared to t the sibling blocks. If no advantages, the child may have conflicting views and GHAST thinks the attack exists.

The attack is ineffective in OHIE [145] and Prism [131] due to their distinguished designs. OHIE achieves a high throughput based on collaborative chains. However, for individual chains, the improvement is not significant since the confirmation of each chain still follows the Nakamoto consensus. A miner has to solve a mathematical puzzle to generate new blocks. Thus, individual chains have low block generation rates, which can resist the balance attack as discussed in GHOST. Prism runs parallel chains with a distinguishable architecture. Generally, the balance attack impedes the confirmation of a block. But in Prism, this block is disassembled into three types of blocks according to its functionalities. These three types of blocks are independent of each other: the proposer blocks, who can pack the transaction blocks, not necessarily wait for voters blocks becomes irreversible. On the one side, a delayed transaction block will not retard the confirmation of leader blocks. An adversary cannot succeed in adding a delay to impede the procedure of other types of blocks. Neither, on the other side, the adversary can hardly partition the network that contains unevenly distributed blocks of all types. Further results on the simulation of balance attacks are provided in [131].

6.3 Splitting attack

In *splitting attack* [22], an adversary can launch double spending between two branches with a high probability. The adversary traverses the network to find two branches/subgraphs whose total cumulative weights are approximative. At this time, s/he attaches a pair of conflicting transactions in different branches to double spend the coins. The conflicting status will maintain for the long term if the adversary continuously sends meaningless units to keep these branches growing evenly. The adversary, thus, can conduct malicious behaviors or gain more profits during this interval. The attack inherently follows a similar principle with the balance attack but still has differences. We observe that this attack is proved to be effective even without any explicit roles of miners. This greatly simplifies the procedures when launching the attacks. An adversary only needs to send massive transactions to the targeted branches, without any further actions to avoid detection from miners.

This attack relies on a simple assumption: an adversary has enough computing power to extend the targeted branches which significantly outpace peers. Thus, the splitting attack targets the naturally expanding systems without any explicitly powerful roles deciding the consensus. Matched systems are mainly *Type I* systems. We take the matured project, IOTA, as an example.

Instances. Tip selection rules determine the properties of Tangle in IOTA [22]. As aforementioned, when the parameter of α approaches 0, the selection rule is uniformly random, while approaches 1, the rule adopts the weighted random walk. A uniformly random selection rule is insecure since the adversary can easily control the branches. Therefore, to avoid the attack, a weighted random walk with the high α value is necessary. Some of the transactions are marked with weights, and attackers can hardly keep the balance between two branches due to the unpredictable distribution of weights. However, a higher α value causes more left behind tips, losing the confirmation rate of transactions. This design limits IOTA in resisting the splitting attack.

A temporary solution has been proposed by their foundation, introducing a central coordinator [181]. The coordinator acts as an authority to periodically (every two minutes) issue milestones for finality. The milestone is a special transaction that confirms the ancestor transactions with a 100% confirmation confidence. Therefore, launching the attack is impractical for adversaries. However, this approach deviates the design principle of IOTA, making it centralized. G-IOTA [86] moves the focus back to the tip selection rules. Increased left behind tips slow down the confirmation rate, since tips stay in low confidences where they are rarely approved by others. G-IOTA accordingly adds one more verification edge for each tip, pointing to the left behind tips. This design helps to increase the confidence of honest transactions fast, while efficiently to detect fake transactions with low confidence for a long time. It remains a higher α value while simultaneously keep the confirmation rate. Further, E-IOTA [87] proposes a parameterized algorithm to adjust the distribution among different types of mechanisms (0 α, low α or high α). This helps to achieve the balance between security and performance. Thus, the splitting attack against IOTA is mitigated by making it possible to choose a moderated α.

6.4 Large Weight Attack

Large weight attack [87] occurs when a *heavy* conflicting transaction invalidates a *recently* confirmed transaction. The term *heavy* represents the values of measurements (score/confidence/weights *etc.*) The *recently* means a transaction is approved by several subsequent transactions, but actually not buried in deep. This is equal to say, the (cumulative) weights of a recent transaction are not too much heavier than the weights of a newly attached transaction. In the large weight attack, an adversary targets a recent transaction,

and immediately generate a conflicting transaction. The adversary improves the weights of the conflicting transactions by methods such as making it approved by lots of meaningless transactions, and append it to the main graph until it becomes heavier than the targeted one. These two transactions belong to different paths. When an honest transaction arrives, it cannot effectively distinguish which path is correct. The transaction can only select the path with more weights. Since the coins in the original path have been irreversibly spent, a double spending in the new path happens when the adversary reuses the same coins.

This attack relies on assumptions in which: a) an adversary has enough computing power to make the conflict transactions heavier than honest ones; b) the consensus is probabilistic without instant finality. Thus, the large weight attack targets the systems with probabilistic-based consensus mechanisms. We provide examples of IOTA and Conflux.

Instances. This attack is effective in the original design of IOTA [22]. Since the tips generated by honest nodes are uniformly distributed in the network, an attacker may need the computing power rate far less than 50% [161]. The temporary solutions based on coordinator [181] shift the IOTA into a completely centralized network, but it is indeed able to resist the attack. The improved approach [114] can also prevent large weight attacks to some degree since the effectiveness of MCMC sharply deteriorate at the margin of the tangle. This increases the randomness of the selection procedure and reduces the probability to unluckily select a heavier transaction. Meanwhile, the effectiveness of the tip selection algorithm is adjustable in E-IOTA [87], which provides a similar resistance against this type of attack.

The attack is ineffective to Conflux [140][141]. The clever point is that Conflux even gets inspired by the concept of weight, proposing its adaptive weight algorithm to make two consensus strategies switch between each other. If an attacker generates and broadcasts meaningless blocks at a high speed to increase one's weights, the system switch to a conservative strategy. This strategy only grants a small portion of blocks with valid weights, whereas others are marked with 0 weights. The attacker cannot control the probability of being selected. Much more power is required if the attacker attempts to have a stable advantage. Conflux also utilizes the adaptive weight mechanism to prevent the balance attack as discussed above.

6.5 Censorship Attack

Censorship attack [196] happens when the adversaries collude enough committee members who execute the consensus. These members may cooperatively frame certain transactions, preventing it from being packed into blocks. Malicious collusion breaks the basic security of a system since colluded members have taken over the system and earn profits according to their willingness. The attack relies on the incentive mechanism. If the members can obtain extra profits through censoring specific blocks or transactions, more members would join the game; if they obtain the profits via honestly mining, these members would keep honest. A positive incentive mechanism, like relating the reward proportional to the honest members [196], will help to maintain a healthy network. Specifically, one can earn a 100% reward if 100% of members are honest; whereas he can only obtain an 80% reward if 20% of members have colluded. Another factor is the attack must collude all targeted members within the time window. If the members leave the committee under the rotation rules, the collusion is useless and costly. These examples prove that the censorship attack is closely related to the behaviors of committee members.

The assumptions of this attack are a) the consensus is completed by a closed committee; b) an adversary has sufficient power/stake to collude a certain proportion of committee members, who may further attract other incomers; c) the committee relies on a relatively static mechanism without dramatically dynamic membership rotation. Thus, the censorship attack mainly targets the permissioned systems whose committees are fixed. Prism is the example.

Instances. Prism [131] provides simulation results against the censorship attack with a tolerable threshold of $\beta = 0.25$, where β indicates the fraction of hash power used for the attack. Prism simplifies the behaviors of attackers to slow down the confirmation by producing empty blocks (proposer block and voter block). Empty proposer blocks may slow down the ordering procedure of transactions since the generation of proposer blocks are delayed. Empty voter blocks may retard the voting procedure which indirectly affects the creation of proposer blocks. Based on simulation results, Prism performs better than the longest-chain protocols *w.r.t* the censorship attack. The confirmation delay of Prism is greatly smaller than delays in longest-chain protocols since the actual delays are merely caused by the insertion of empty blocks. This is the key point to resist the attack, where each decoupled blocks would not hinder other types of blocks. In contrast, the delays in the longest-chain protocols would impede all the procedures.

6.6 Replay attack

Replay attack [197] refers to stealing users' coins by replaying transactions multiple times. The key to performing the attack requires an adversary to reuse the address of the victims. Reusing the same address makes certain transactions repeatedly confirmed and all the addresses following this vulnerable address become vulnerable. Adversaries may drain the funds from all transactions associated with this vulnerable address. In case of the funds in this address is insufficient, the adversary firstly tops up the address, and then replays the transaction to steal more funds until using them up. The replay attack is mainly used for the systems based on the UTXO model, because the remaining values have to be instantly transferred from the current address to another new address, rather than directly being removed as it in the account model. Besides, two types of variants are identified. The first one is *brute force*, where an adversary tries to guess the seed. But it cost massive computing power which may not be worthy of it. Another type is to analyze the past transaction to find a potential input address with remaining values. All these variants still rely on reusing addresses.

The assumptions of this attack are: a) the remaining values of one address has to be transited to another address; b) the address can be reused for multiple time through some technique. Based on this, the replay attack targets the systems which adopt the UTXO model as their data structure. We take the IOTA as an example.

Instances. IOTA [22] relies on *bundle* [198] to finish the token transferring. The bundle is a virtually top-level construction used to simulate accounts for users. Each bundle links related transactions

to complete the operations through a set of addresses generated by the same seed from the user. Specifically, the remaining values of transactions in the bundle need to be transferred from the current address to a new address. These two addresses are linked by the seed. The user who holds the seed can control transactions that relate to him. Replaying attack is feasible in IOTA since the transaction bundle is inherently based on the UTXO model. Joseph [197] shows the possibility of the attack, while Roode *et al.* [72] provides a practical method to conduct the attack. The method modifies the functions (mainly `addRemainder`) of IOTA's API, enabling input address reused. The modification removes the procedure of generating remaining funds such that the funds can stay on the input address. Fortunately, this attack can be mitigated. Suggested by [197], each signed transaction bundle should be tracked via its unique hash, and each subtangle only permits one bundle. As a counterpart, this method increases the overhead.

6.7 Sybil Attack

Sybil attack [199] is a common attack in the P2P network which is also adaptable to the blockchain network. An adversary may generate multiple pseudonymous identities to conduct malicious behaviors, such as disconnect the channel between nodes, or even taking control of the whole network. Specifically, an adversary may add a large number of pseudonymous identities in the blockchain network to participate in the membership selection. If one of them is luckily selected to be the committee member, s/he can arbitrarily generate blocks that benefit himself. The fake members can efficiently forward the block generated by adversaries whilst abandon the genuine blocks from users. As a result, the adversary who controls the network finally reduces the overall throughput and obtains extra rewards. The more fake identities are created, the larger chance they are selected. The key to successfully launch this attack is to generate a large number of identities with low/zero costs. Then, a malicious node will use these identifies to gain disproportionately significant influence.

The attack relies on the assumptions: a) generate identities without any cost; b) the nodes have enough power to efficiently create identities. Sybil attack affects the network by its generated massive identities. Current blockchains that adopt PoS/PoW consensus effectively avoid this vulnerability since becoming a block packager (membership selection) must simultaneously solve a lot of puzzles (block production). Therefore, this attack mainly targets the systems whose committee selection and block production are separate. We take the Blockclique as an example to show how to avoid this attack in DAG systems.

Instances. Blockclique [180] indicates that its protocol relies on a Sybil-resistant selection mechanism. Achieving this mechanism needs to meet two requirements: a) a valid membership of nodes must be obtained with a sufficient cost, and b) the membership in the next round cannot be guessed in advance. The key of the first requirement is to explicitly select a node based on the *proof of ownership of a resource*: nodes have to spend some resources and add certain delays each time participating in the node selection. The more costs or mortgages are spent, the more reputation a node will obtain. This increases the confidence of a node, decreasing friction when communicating with others. The core of the second requirement is to guarantee randomness without leakage. Blockclique utilizes an oracle to describe the random selection of nodes. All nodes need to consult this oracle to confirm their membership in a certain slot. To meets these requirements, Blockclique gives two adaptable mechanisms: the PoW-based membership selection as in OHIE [145] or ELASTICO [200], and the PoS-based selection as in Tezos [201]. OHIE utilizes the residue of hash to decide which thread it switches to. ELASTICO employs PoW to generate identities and then select the membership randomly from those identities. Nodes in Tezos are randomly selected with a probability proportional to their stakes. All these selections prevent the adversaries from maliciously spawning massive nodes.

7 PERFORMANCE

In terms of performance, we consider *throughput* (*i.e.*, the maximum rate at which values can be agreed upon by the consensus protocol), *scalability* (*i.e.*, the system's ability to achieve greater throughput when consensus involves a larger number of nodes) and *latency* (*i.e.*, the time it takes from when a value is proposed, until when consensus has been reached on it). Here are the details.

- ***Throughput*** (λ) is the maximum rate of confirmed units over the network. It is affected by various factors, including the limitations of bandwidth, the design of consensus protocols, and the fraction β of (hashing power of) the adversaries. The throughput in DAG systems calculates the confirmed total units during a specific time frame, no matter they are collected from one single chain or multiple chains. We follow the classic measures called *TPS* (Transaction Per Seconds).
- ***Latency*** (τ) mainly consists of two parts, the unit *propagation time* and *confirmation time*. The propagation time is the length of time that a unit reaches its destination. It is closely related to the parameter *coverage*, used to describe how many nodes received these broadcasted units. The confirmation time represents the length of time that a unit is deemed as confirmed. For instant confirmation protocols (*e.g.*, BFT-style), the consumed time is static on average. For delayed confirmation protocols (*e.g.*, NC-based), the time is unpredictable. The probability ε of a unit is removed by its child will drop as the chain/graph grow. When the unit is buried deep enough, an approximate time is obtained.
- ***Scalability*** (ϕ) shows the ability of a system to handle a growing amount of units when adding a large number of nodes. The assumption of consistency is the leading factor that influences the property. The strict consistency (which requires total linear order of the units) will significantly retard the performance since the complexity of consensus increase with the enlarged committee. In contrast, a weak consistency can greatly scale the blockchain due to its tolerance of forks. Other factors including the design of consensus protocols, the fraction of malicious nodes, *etc.*

General Analysis. Performance is the bottleneck of Bitcoin due to its security-limitation. Simply increasing the mining rate can improve the performance but with the expense of decreased security. Suppose the block creation rate of an attacker is greater than the growth rate of the main chain, the malicious chain will eventually outpace the honest chain. Thus, increasing the mining rate

Table 4: Attacks Analysis

	Behavior	Assumption	Scope	Instance
Parasite Chain Attack	Adversaries secretly generate a subgraph to replace the honest subgraph with his prepared ones for more profits.	- Enough power to generate a parasite chain outpacing than peers - No instant finality	Probabilistic consensus	IOTA Spectre
Balance attack	Adversaries partition the network into balanced subgraphs and start mining on one subgraph to invalidate another that has a deal with merchants.	- Split/partition the network - Secretly wander across subgraphs - Enough power to extend subgraphs	PoW-based protocols	GHOST, Conflux, OHIE, Prism
Splitting attack	Adversaries insert a conflicting transaction into two balanced branches to double spend the coins.	- Enough power for balanced subgraphs - No instant finality	Probabilistic consensus	IOTA
Large Weight Attack	Adversaries creates a conflicting transaction with high confidence to invalidate a recently confirmed transaction.	- Units heavier than others - No instant finality	Probabilistic consensus	IOTA Conflux
Censorship Attack	Adversaries collude enough committee members who execute the consensus to take over the system and earn profits by preventing certain transactions from being packed into blocks.	- A closed committee - Collude the majorities - Static committee selection mechanisms	Permissioned systems	Prism
Replay attack	Adversaries reuse the same address of the victims to steal users' coins by replaying transactions multiple times.	- Balance has to be transited - Reuse addresses	UTXO-based systems	IOTA
Sybil Attack	Adversaries create massive pseudonymous identities to participate in the membership selection (to become fake miners) to conduct malicious behaviors.	- Generate identities without any cost - Enough power to create identities	The committee selection and block production are separate	Blockclique, Nano

while maintaining security is critical for improving the throughput and latency of protocols. Current (leading) systems avoid the aforementioned limitation in three independent ways as follows.

- One line of work (GHOST, Conflux) provides the solutions by *maximizing the mining rate until reaching the security-limitation*. These solutions follow the main architecture of Bitcoin, but with the modifications of adopting more sophisticated extension rules to solve forks. The systems mainly belong to *Type V/VI*. These systems have to extend graphs and sort units in the main chain. Such designs confront a similar bottleneck with classic blockchain systems. Improving performance relies on decreasing the confirmation time of the main chain. The time cost from conflict solving and unit sorting limits the upper bound of throughput.
- The second line of work (Prism, OHIE) aims to *horizontally scale the blockchain by enabling multiple chains processed in parallel*. The system adopting this type of method generally has a fixed security threshold. Within this threshold, the throughput can maximally approach the communication-limitation. Conversely, once exceeding the threshold, the system is limited by its security bound. Improving scalability without undermining safety is the key point in this line of studies. This design pattern mainly appears in *Type III* and *Type IV*. The systems are based on structured DAGs and each individual chain speeds up simultaneously.
- Another line of work (IOTA, Avalanche, Spectre, Phantom) *reconstructs the systems to reach the (physical) communication-limitation*. The approach greatly modifies the original designs, decreasing the confirmation time as well as improving the scalability. However, as the sacrifice, the systems inevitably either weaken the security promise or increase the system complexity. Such limitations arise from excessive random forks due to their unstructured networks when the mining rate is increased. These systems frequently appear in *Type I* and *Type II*.

We further capture several experimental testing results from the corresponding literature and list them in Fig.31. To be more specific, the performance of a system can be measured in total TPS, the multiplication of individual chain throughput, and participated nodes. We observe that existing DAG-based blockchain systems cannot achieve both of them at the same time. Suppose these two factors sit at the two opposite ends of the spectrum. Frankly speaking, it is not clear what the optimal solution is for the sweet spot when considering trade-off between scalability and throughput. The bottleneck may be limited by security consideration or physical boundaries. We provide further detailed analyses on several leading systems.

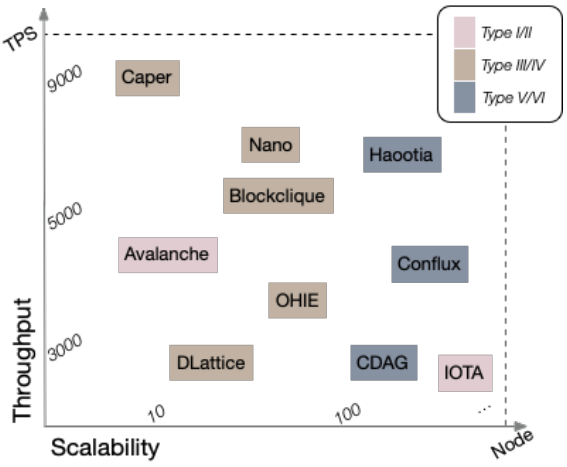

Figure 31: Throughput and Scalability

Instance Analysis. We provide detailed analyses of mentioned instances. Due to the differences between experimental settings and hardware devices, we focus on the reasons to gain high performance and their potential sacrifices, instead of quantitative outcomes.

GHOST [45] follows the model of Bitcoin, but with the modifications on chain selection rules. The protocol extends the chain by selecting the *weightest* subtree, instead of the *longest* subchain. It greatly changes the way to organized blocks. Unlike Bitcoin's linear structure, GHOST organizes blocks in a tree structure which improves the utilization of mining power under high contention. This design enables GHOST to significantly improve the block generation rate, and decrease the confirmation time of blocks. A high generation rate leads to unsolvable forks in Bitcoin, whereas a large proportion of the forks are tolerable in GHOST. As proved in [45], when the rate moronically increases, the security threshold of Bitcoin sharply decreases, whilst the GHOST protocol maintains stable. Forks frequently appear in a high block generation rate, but GHOST is more resilient and scalable than Bitcoin.

Conflux [141] inherits the design of GHOST to achieve high performance without undermining security. Standard NC-based protocols reach a total linear order of transactions while confirming blocks. The key insight of Conflux is to decouple the confirmation of blocks against transactions. Differing from Bitcoin and GHOST, Conflux defers the steps of the transaction sorting, making it processed after the block confirmation, rather than at the same time. The order of transactions can be deterministically derived from the agreed order of blocks. The dependency between these two components is thus changed. Each time a block only needs to concentrate on a gross-grain sorting of blocks, without having to simultaneously consider the fine-grain sorting of transactions. Besides, Conflux enables each block to reference multiple edges at each time, which allows concurrent transactions and blocks. Conflux also stresses its adaptive weight mechanisms to switch the strategies between a fast model and a slow mode. This design smoothly balances security as well as performance.

Spectre [24] and Phantom [80] embrace the DAG structure to support faster block generation and larger block volume. The improved performance of Spectre comes from two aspects: a) The system structures blocks to form a topological network. Transactions can concurrently attend to the network, making the system scalable. b) Increasing the block generation rate contributes to performance. This is due to the fact that Spectre only requires a pairwise ordering between two blocks, avoiding the barrier of conflicting states among many blocks. Similarly to Spectre, blocks are created and attached concurrently in Phantom. The main difference is that Phantom requires a total linear order of blocks and transactions. This makes Phantom have to maintain a trade-off between security and performance. Simply increasing the block generation rate will unpredictably affect several associated factors. Thus, the confirmation time of transactions is much slower than that in Spectre, but still faster than any NC-based blockchain systems under the single linear chain structure.

IOTA [22], adopting the third metho, both decrease the confirmation time and improve the system scalability. It relies on a simple and intuitive tip selection rule to extend the graph without any further limitations. The fast block creation rate directly improves the confirmation rate for pending transactions. A peak rate (> 400 TPS) has been observed with the confirmation coverage of over 99% [41]. It is sufficient for most application cases. Further, we capture two features with respect to the performance of IOTA. a) Theoretically, the performance is merely limited by the physical hardware, including bandwidth and propagation delay. The Tangle can grow without hindrance in which forks are tolerable. b) The confirmation time of transactions significantly varies according to their confidence. The higher confidence a transaction initially obtains, the faster it can be confirmed. Uneven distribution of confirmation time is formed since most of the transactions follow the biased incentives. As a sacrifice, the design of IOTA inherently weakens the strong guarantee of consistency and security. IOTA has to (temporarily) rely on a centralized Coordinator to periodically take snapshots of the history to balance stability and performance.

Avalance [130] redesigns the consensus protocols by leveraging network sampling. Similar to IOTA, Avalanche achieves high throughput and scalability with the expense of undermining strict consistency. The high performance derives from three aspects: a) Leaderless BFT protocols in Avalanche avoids the congestion and vulnerability of the centralized authority; b) The system only maintains correct pairwise orders rather than a total linear sequence of transactions. c) The complexity of handling a message is fixed ($O(k)$), free from the expansion of the network. Besides these three intuitive reasons, an additional factor is the fraction of Byzantine nodes in the network. The probabilistic guarantee of its safety will significantly degrade when Byzantine participants exceed f due to the exponential effect from the repeated sampling. Keeping the fraction at a low level enables the subsampled voting mechanism to reach the consensus more safely and efficiently.

Prism [131] deconstructs the Nakamoto consensus into basic functionalities and reconstruct its new system by scaling up these functionalities. The bottleneck is limited by physical conditions, rather than security requirements. The key point is that all functionalities are decoupled and independent. This design greatly improves the performance in the following aspects: a) The transaction blocks

are instantly confirmed by the powerful leader blocks without delay. The confirmation does not rely on any intensive-resources like computing power or stakes, which enables fast execution. b) Th leader block confirms a list of possible ledgers instead of a unique ledger, making non-conflicted transactions simultaneously processed. c) The confirmation of a leader block will not be hindered by the states of voter blocks. Once the leader block receives a majority of votes from voter blocks, it becomes permanent without having to worry about situations of voter blocks. Prism can be deemed as the first NC-based protocol (*w.r.t* the leader chain) that breaks the confirmation barrier of transactions.

OHIE [145] scales NC-based protocols by processing chains in parallel. The system achieves high performance as well as maintaining a linear order of transactions thanks to the design that speeds up almost every procedure. Specifically, OHIE makes improvements in the following aspects: a) A pre-allocation of transactions into individual chains avoid the duplicated confirmation and potential conflicts. Assume a total of k transactions and n nodes, each chain just needs to take $1/n$ workload to processes k/n transactions. b) The system applies NC-based extension rules to each single chains, rather than the whole system. This is essential to overcome the bottleneck because much of the time is wasted under high contention of forks. Instead, OHIE sorts the blocks by comparing their intrinsic parameters (positions) at a high speed. 3) OHIE decouples the functions of confirmation and sorting where sorting blocks happens before the confirmation of blocks. The sorting relies on the location of a block, while the confirmation is based on its proposed *confirm bar*. This design provides an additional benefit of scalability. Unlike BFT-style protocols, newly joint nodes (with legal permission) will not increase the burden of existed ones and the system can still reach the peak performance. The limitation of OHIE comes from its sorting algorithm.

Insights: Based on previous discussions, we conclude some insights. Since the design of each DAG system varies from one to another, we capture several common features as follows.

- ◇ *We can hardly improve the scalability and throughput at the same time under the assumption of strict consistency.* The more nodes are involved, the more time will be spent on decisions. Even though each type of systems are radically different, they still have to reach the trade-off between security and performance to a certain degree.
- ◇ *To achieve strict consistency, the sorting algorithm might become the bottleneck in DAG-based systems.* The computing complexity in sorting algorithms exponentially increase with the participated members. One particular case is that the committee member only has one member (might dynamically change) – a leader who can decide the sequence, such as Vite [139], Jointgraph [171]. This design maximally decreases the confirmation time but results in centralization.
- ◇ *Weakening the strict requirements of consistency enables high scalability and fast confirmation.* Systems like IOTA [22], Nano [50], Spectre [128], only guarantee the partial/pairwise order of the transactions. This is enough for asset transferring between two/multiple parties, but cannot support strict state transitions like smart contract components.
- ◇ For the NC/NC-variant based DAG systems, we observe that *the depth d only has impacts on the confirmation latency, rather than the throughput and the propagation latency. Depth* is used to measure how deep a block is buried in the chain.
- ◇ *Theoretical analysis of performance usually adopts Nakamoto's model* [178], such as the analysis in GHOST [45], Prism [131], OHIE [145] Conflux [140]. Unfortunately, only a handful of them provide experimental evaluations.

8 DISCUSSION

In this section, we discuss the a) ways to integrate each component (structure, consensus, properties) and their impact on each other; b) differences between DAG technique and existing scaling approaches; c) related components with their impact on DAG systems; d) challenges surrounding DAG-based blockchains.

8.1 Intra-Component

This subsection explores the mechanisms, features and differences of each sub-component inside each component. We discuss two physical components *structure* and *consensus mechanism* and one virtual component *property*.

Structure defines the way units are recorded in DAG systems. It consists of two aspects, namely *data structure* and *network topology*. Traditional blockchain systems neglect the second factor since they rely on the linear-chain topology as default. In contrast, graph-based topology in DAG systems is the most distinguishable feature. Here, we provide more discussion.

Data structure. The metadata stored on the ledgers can be structured in either transactions or blocks. A major difference between these "containers" is their permissions to operate the ledgers. This includes the rights of *read* and *write*. *Read* means a node can access the states in the ledger and obtain the messages to generate his new transaction without changing any previous states. *Write* means the node can change the states of the ledger by adding new content. In the transaction-container systems, all participants can both *read* and *write* the data into ledgers. The nodes who receive legal transactions will directly append them into their ledgers. However, in block-container systems, the committee members (miner/validator/*etc.*) have the rights of *read* and *write*, whereas the ordinary participant can only *read*. Transactions sent by the ordinary nodes require to be packaged by the committee nodes and then written to the ledger after the consensus. Ordinary participants cannot directly affect the status of the account. Additionally, the difference also lies in its role of *filter*. The committee nodes can filter malicious transactions to some extent. A double spending transaction will be abandoned directly, rather than being appended into the ledger. This effectively resists the flooding attack and replay attack.

Topology. Topology refers to the graph formed by the units and their relationships. It also describes the way to organize each unit in the network. Three design patterns are identified in Section 3, namely *divergence* ($\widehat{D}$), *parallel* ($\widehat{P}$), and *convergence* ($\widehat{C}$). The topology cannot determine the final properties. It only shows the visualized graph made by edges and vertexes. Additionally, the topology can indirectly reflect the complexity of the protocols. $\widehat{D}$ type contains the most flexible systems without strict/uniform consensus mechanisms. Transactions and blocks can arbitrarily be dispersed

in the network. Only a simple rule (TSA/RTA/GA/Sampling) is applied to each system. $\widehat{P}$ type includes the systems with a group of nodes who separately maintain their chains/accounts. The complexity depends on their aimed properties, such as the linear order. An additional sorting algorithm is introduced to the system for the final sequence. $\widehat{C}$ type generally consists of two main steps, from the main chain and determine the sequence. The main chain both records the history as a trusted snapshot chain and solves the conflicts among blocks and transactions.

Consensus mechanism defines the way to reach the agreement. Traditional linear-based blockchain systems decouple the consensus into two main questions: *who* conducts the consensus and *how to* operate the consensus [15]. We follow this category (see Section 4) with additional details as follows.

Openness and *membership selection* answer the question of *who*. In blockchain systems, a group of nodes that participate in the consensus procedure is called the committee. These sub-components define the rules on how to select a set of trustworthy nodes to form the committee. The committee is essential for successfully running consensus algorithms since it improves the security threshold of the systems. Specifically, for an open environment (permissionless), nodes compete for winners according to their resources, like computing power, stake, reputation. For a closed environment (permissioned), the member is generally assigned by the developer team. Adversaries cannot become legal members if they have not invested in large amounts. Even though an attacker temporarily holds sufficient resources, the advantages cannot last for the long term. For example, IOTA [22] only utilizes the PoW as the anti-spam tool, rather than the rules of membership selection. An attacker may easily launch the parasite attack and splitting attack (see Section 6). Thus, applying the coordinator to the IOTA network is evidence of improving the security threshold. Similarly, assigning several authority nodes as the legal members also exclude unfamiliar nodes outside the committee. Besides, the small size committee enables the systems to run optimally with lightweight payloads. This is especially critical for parallel-chain based systems (*Type III, Type IV*). The more nodes are involved, the more messages are broadcast.

Unit allocation, *unit positioning*, *extension rule*, *conflict solving* and *featured technique* answer the question of *how*. Classic blockchain systems focus on the procedure of extension rule where the chain only needs to consider how to select its child. The rule is straightforwards since the metric of child selection is based on one dimensional parameters, such as length (Bitcoin [1]), weight [45], age [202]. In contrast, DAG-based systems are complex. Firstly, the role of the units is uncertain. A unit might be a transaction, an event, or a block and the role of a block can even be decoupled into different functionalities (see Prism [131]). Secondly, locating a unit, as the pre-step for sorting algorithm, is hard in the graph-based network. A well-structured DAG (such as *Type III*, *Type IV*) can position a specific unit by at least two parameters, (i, h) in Table.3. The sub-component *Unit position* helps to mitigate this issue. However, an unstructured DAG (*Type I, Type II*) cannot precisely locate their units. Thirdly, each DAG system has its designs with customized aims and properties. Not all of the systems require strict consistency or total ordered sequence. This leads to different extension rules in multidimensional metrics. We separately use *extension rule* and *conflict solving* to describe their consensus approaches. Extension rule mainly refers to the principles of tie-breaking, while conflict solving represents the way to sort the selected units into the final sequence. The *featured technique* emphasizes the key techniques adopted by each system. A complete consensus mechanism in DAG systems might include all of the sub-components or several of them.

Properties are not physical components (like structure and consensus). Instead, properties describe the final goals of each system. It is not standalone since the aimed properties determine how to design the algorithms and integrate them together. Three properties are identified in Section 3, namely *consistency*, *order*, and *finality*. Consistency describes whether a state is globally viewed among different ledgers; order shows how each unit is ordered, and finality represents the possibility that a ledger can be reversed. These three factors are orthogonal in most cases but might have overlaps in a few cases. For example, Vite [139] consists of multiple parallel chains and one snapshot chain. If we watch from the whole system, the consistency is not achieved since the ledgers in each node have totally different views. In contrast, if we watch from the snapshot chain, consistency is achieved. Such conflicts are caused by the role of its main (snapshot) chain. In most cases, the main chain records the full history and simultaneously conducts the consensus. The decisions will be synchronized to each distributed nodes, and they can reach the same view of states. However, if the main chain only records the history without further consensus steps, common nodes cannot obtain any synchronized states. In this situation, we measure the consistency in the view of its main chain.

8.2 Inter-Component

This subsection explores the connections between individual components and their impacts on each other. Two physical components *structure* and *consensus mechanism* and one virtual component *property* are included. Here, we provide more details.

Structure and Consensus. The structure defines the way transactions are recorded in the systems. It greatly impacts the consensus of the unit types they agreed upon. In our classification, two major types of units are identified: *transaction* and *block*. If the system is directly based on transactions, the consensus tends to be simpler than that based on blocks. The former only requires the rules of conflict solving and unit sorting, whereas the latter needs additional procedures such as membership selection, the configuration of committees, *etc.* Conversely, the consensus also diversifies the types of the structural design. For example, an additional snapshot block in Vite [139] is introduced into the system due to the design of consensus. We ignore the classification of block types at more specific levels, such as vote/propose/transaction blocks in Prism [131]. We consider they are encompassed in the consensus procedures.

Consensus and Property. The Properties indicate the goal of consensus, and conversely, consensus algorithms are the way to achieve properties. Traditional blockchain systems do not often mention this virtual component since they attempt to achieve strict consistency, or the total linear order, as their default requirements. In contrast, DAG-based systems relax such requirements, letting systems achieve partial consistency or stay in topological order. Thus, the consensus algorithms under different properties vary from system to system. The consensus procedures and complexity

to achieve the total linear order are much more complicated than staying in the topological order. But the latter sacrifices security and compatibility to some degree.

Structure and Property. Properties indirectly impose the requirements on the structure design. For example, a unit to be deemed as valid or confirmed requires the high confidence to prove it is attached by sufficient child units. This makes the unit equipped with a counter to measure confidence. For the systems based on transactions, the counter can only be initiated as an additional field: weight, vote, *etc.* In contrast, for the systems based on blocks, the counter may consider more options like the block volume or the block size, which are used to show the collected/verified transactions in every single block. Also, some powerful blocks may instantly have the decisions on contained transactions, like the propose block in Prism [131].

8.3 Differences with Parallel Approaches

The performance and scalability can be optimized by many techniques. We conclude the most prevailing approaches and highlight their differences compared with DAG.

Sharding technique [11] splits the transactions into disjoint shards to enable them processed in parallel. DAG approach differs from the sharding technique in the following aspects: a) The systems adopting the sharding approach rely on strict consistency. In contrast, strict consistency is not necessarily for DAG systems. b) Sharding-based systems still utilize the linear-chain structure to maintain their ledgers for final states. The separated zones have to reach an agreement in each epoch across multiple shards, which makes c) the multiple-committee consensus becomes essential in the sharding technique. The key idea of such consensus protocols is to assign the nodes to committees in a non-deterministic way, preventing the adversary from threatening specific members. Conversely, the DAG systems are generally based on single-committee consensus. Some studies [16] claim that single-committee consensus is not scalable where newly participated nodes decrease the throughput. This is only one side of the coin since the assumption of consistency is neglected. If the system only requires partial consistency, like Nano [50], a single-committee will not become the limitation for scalability.

Layer2 technique [13][203], refers to the solutions that process certain transactions outside of the main chain, but the consensus of theses transaction still relies on a parent chain. The parent chain is a traditional linear-based blockchain, selectively recording the important transactions (*e.g.* balance, final state). Compared with DAG, this technique is inherently an upper-layer protocol without changing the structure or undermining the properties of its parent chain. Layer2 technique cannot maximally improve the performance since the consensus is still limited by its linear-based parent chain. The bottleneck is significantly affected by its main chain. Another critical problem is to reliably guarantee the consistency between off-chain states and on-chain states. Relying on authority nodes for endorsement or charging the deposit in advance might increase extra costs either computationally or economically.

Sidechain technique [14] deviates from the layer2 technique in whether they have own consensus algorithms. The sidechain is a separate blockchain that processes transactions individually. It interacts with the main chain and tokens can be transferred between them. Several DAG systems (such as Vite/Parsec) get inspired by this design to some degree, which introduces a hybrid architecture with one main chain to sort the units and multiple ordinary (side) chains to process transactions in parallel. However, a major difference is the coupling between the main chain and side chains. In DAG-based systems, the main chain and side chains are closely integrated. The states (metadata) are recorded on all involved side chains and the consensus across these chains (ledgers) is then finished with the help of the main chain. In contrast, the main chain and side chain are completely decoupled in this technique. The side chain is merely an auxiliary chain, providing a very small proportion of information to the complete system.

Heterogeneous structure represent a vertically structural shift. The approach changes originally homogeneous blocks into heterogeneous blocks by adding new block types. It assigns the blocks with different functionalities, generally, including two types: the *keyblocks* used to conduct consensus and the *microblocks* to vote for leaders and carry transactions. This line of studies include Bitcoin-NG [204], Fruitchain [185], RepuCoin [205], ByzCoin [206], and ComChain [207]. All these systems improve the throughput of performance without terribly undermining the security promise. The most significant difference is that: DAG approach inherently is the horizontally scaling solution that aims to enable transactions processed in parallel. Several DAG-based systems directly adopt the concepts of this technique, such as Prism [131], to decouple the functionality of blocks; while some systems indirectly utilized the concept, such as Conflux [141]/Vite[139], to organize the keyblocks in a separate (main) chain for consensus.

Hybrid consensus solution refers to the systems which combine two or more consensus protocols. These systems attempt to abstract the advantages of each system and integrate them into one protocol. Thunderella [208] replaces the complex process of view-change in PBFT with the Nakamoto consensus. This makes the system smoothly switch between the optimistic conditions and the worst-case conditions. In optimistic conditions, Thunderella provides a high throughput since the adversary is weak, whereas, in the worst-case conditions, it falls back to the NC protocol for the conservative guarantee of security. RepuCoin [205] utilizes the proof of reputation mechanism to increase the security threshold while uses the BFT protocol for fast state agreement. Other systems includes Tangaroa (Raft+BFT) [209], Tendermint (PoS+PBFT) [210], *etc.* However, these systems only optimize the performance at the consensus level, which cannot overcome the bottleneck from the linear-based structure or topology. DAG-based systems also adopt the concept of this approach to construct their integrated components, especially how to combine the procedure of committee formation and agreement decision.

Other technique [17] includes on-chain solutions that proposes modifications to the blockchain protocols. Increasing the size or improving the volume of blocks is the key feature. BCH [18] modifies the hardcoded parameter, adjusting the block size from 1M to 8M. Larger blocks increase the total throughput and reduce transaction fees. SigWit [211] splits a classic block (1M) into the associated blocks: a base transaction block with a size of 1 MB and an extended block with 3 MB. These blocks individually take different functionalities. However, these systems typically require a hard/soft fork

from the original blockchain, which costs much time for negotiation between their communities. Compared to DAG systems, this approach merely changes one of the components of blockchains, rather than any structure or topology. The performance bottleneck still depends on its base blockchain.

8.4 Research Gaps

This subsection provides a handful of issues that have not been well addressed in current DAG-based blockchain systems. We hope that the communities and following researchers can tackle the open problems as we listed.

Absent Details of Committee Configuration. Maintaining liveness and security in committee configuration is an open and neglected issue. We observe that current committees are formed either statically or dynamically. Specifically, static committee configuration lacks liveness, making systems vulnerable towards attacks like DDoS attack, Sybil attack, and censorship attack. Dynamic committee configuration improves security by raising the bar for attacks but brings new challenges on how to maintain liveness. It includes the frequency and fraction of committee membership (epoch, dynamism). The property directly relates to the committee's security. However, the systems which seriously discuss or comprehensively analyze the issues only make up a minority.

Trade-off. Blockchain systems cannot improve the performance, scalability, security, decentralization, and strict consistency at the same time. For example, we observe that some systems, like IOTA [22], Graphchain [79], improves the performance and scalability at the expense of security and consistency. While some systems, such as Prism [131], OHIE [145] ensures strict consistency, but the scalability and performance are sacrificed as a balance. In traditional distributed systems, FLP Impossibility theorem [212] and CAP [213] theorem help us understand the limitation of some factors. However, in DAG-based systems, or even blockchain systems, there is no theorem or experience to clarify the boundaries of each factor, the way of reaching a balance between multiple factors, or even the range of factors that are involved.

Privacy Issue. Privacy in classic blockchain systems [8] is a complicated issue. Generally, the privacy of blockchain systems includes two categories: *anonymity of identities* and *privacy of metadata*. Firstly, Bitcoin allows users to generate multiple addresses. Adversaries can obtain the link between these addresses and user entities by collecting massive historical transactions and analyze the relations to pursue profits. Most DAG systems following this concept will confront the same threats. Secondly, the metadata on the ledger is shown in plaintext as default. Adversaries can trace any interested transactions (*e.g.* with large balance) and conduct harmful activities by locating their associated accounts. No (effective) privacy-preserving solutions have been applied to the DAG-based systems up to now so far.

Informal Model. Many of the aforementioned systems are informally presented and lack formal models either on security or properties. This leads to confused proofs or discussion when they claimed to be secure. Several systems are presented in a formal manner, formalizing the models consistent with their properties. More specifically, although the models vary from system to system, we still list them as the educational samples. These systems are shown in Table.5. Besides, several studies further provide analysis with formal models, including topics on properties [26] or performance (of parallel-chain systems) [83] We encourage future work could be conforming to this format, enabling the proposed systems to be discussed thoroughly and critically proved.

Table 5: Systems with Formal Models

	Adopted Model
IOTA [88][157][158]	Markov Process
Avalanche [130]	Continuous-Time Markov Process
GHOST/Inclusive [45][46]	Nakamoto's
Spectre/Phantom [80]	Self-defined
Prism [131]	Nakamoto's (Backbone [178])
OHIE [145]	Nakamoto's (Backbone [178])
Conflux [140]	Nakamoto's
Haootia [82]	Crypto-style proof
Informal	[79][24][50][81][29][48][171][147][167] [123][139][170][179][132][148][136]

Informal models lead to non-unified properties in different systems. We mention several types of properties that defined in concurrent literature, such as *effectiveness*, *atomicity* and *timeliness* in [19]; *efficiency* in [26]; *liveness* and *correntness* in [3]; *persistence* and *liveness* in [178]; *etc.*

Absence of System Setup. Setup configuration refers to the information that can be available at the onset of the protocol to each participant. Three types of setup configurations are introduced in [20], namely *no setup*, *public-state setup* and *private-state setup*. Most NC-based systems (Bitcoin, *etc.*) rely on a global-viewed genesis block as the initial state, while a number of systems (such as [214]) need to have some pre-existing configurations (such as the knowledge of a common reference string [CRS], or a public-key infrastructure [PKI]). In current DAG-based blockchains, some of the systems rely on the genesis block, mainly *Typel/II/V/VI*, whereas some systems simultaneously initialize several chains in parallel, mainly *Type III/IV*. It is unclear how these parallel chain based blockchains bootstrap the systems.

Absence of Incentive Mechanism. Three major goals of incentive mechanisms are a) letting more nodes get involved in a committee to participate consensus; b) encouraging honest behaviors of committee members; and c) directly affecting consensus mechanism, such as Graphchain [79]. There has been little investigation into how to build incentives for increased participants. The absence of incentives makes it hard for committee members to maintain the system stable and reliable since no laws guarantee they never get compromised. Although several DAG systems claim that *feeless* is one of the most outstanding advantages in DAG, they still utilize additional techniques as the complementary mechanism. For example, IOTA [22] introduces a trusted authority to maintain stability.

Unknown Compatibility. Although this paper provides insights into physical components (structure, consensus) and featured techniques (see Section 4), there are still a large number of related components and techniques that have significant influence on their designs/operations. Current studies lack corresponding description, and we provide some instances as follows.

- ***Smart contract*** [215] is an essential component to achieve state transition in classic blockchain systems. However, none of state-of-the-art DAG systems have implemented the mature smart contract. This is mainly because a complete state transition highly relies on the total linear order of units, ensuring the correctness and integrity of states. The systems without linear order can only encompass auxiliary components. For example, IOTA provides a decoupled upper layer solution [92] as the external component.
- ***Sharding*** [16] splits transactions into separated zones. Several DAG systems (Blockchique [180], Eunomia [146], Dexon [134]) get inspired from the sharding technique to allocate the transactions into different chains in advance, preventing potential conflicts during the consensus. However, DAG systems only capture the simplest and most straightforwards concept from sharding. The issues such as how to fairly and randomly allocate the transaction, how to maintain the separate chains relatively balances, are still left as the gaps.
- ***Layer2 technique*** [13] is an upper layer solution for linear-based blockchains. As discussed in Section 8.3, layer2 technique attempting to solve performance issue share lots of similarities with DAG approach in the perspective of system design. It is interesting to explore whether the layer2 technique can support more applications for DAG systems, especially the scenarios having to manage both off-chain and on-chain transactions.
- ***Cross-chain*** [19] enbales the chains that are heterogeneously structured able to communicate with each other. The usage of this technique requires powerful tools such as the global clock. Traditional blockchain systems achieve the goal either relying on a third party as an external clock or depending on chain-dependent time definition, such as the block generation rate. The tools are relatively easy to implement on their (unique) main chains. In contrast, DAG systems can hardly achieve the regiments for cross-chain due to their inherent complex situations: they create many forks (*Type V/VI*), coexist multiple chains (*Type III/IV*) or even without the main chain (*Type I/II*). The difficulty to synchronize the information, both internally or externally, hinders the application of across chains in DAG systems.

Whitepaper Information. Whitepapers were the earliest documents to express the idea and proposals from developers and technicians. These documents encompass brief models, related components, and, more importantly, the developing plan, which targets future investors. In this paper, not all the discussed DAG systems are written in the form of academic papers. For those non-academic documents, we only select and discuss the systems (see Table.3) with relatively formal documents, which clearly presents the design details and technical models consistent with the literature. Besides the selected systems, we also summarise ever-existed projects as shown in Table.6. We have reviewed these systems in detail, most of which follow similar models and designs as discussed in the aforementioned section. We only list them with project titles. The latest information is updated here.

Deployment Difficulty. Deploying proposed DAG-based systems is not easy for developers. Most implemented systems, like [50][180], only present the designed protocols with incomplete evaluations, rather than releasing their source codes. A developer can hardly learn of the full logic by conditional access to the repositories. Many systems [22][139], with accessible repositories, do not provide proper documents on guidelines or deployment. Other systems, such as [48], are developed by official teams, but their design is protected by the patent. The current status shows the widespread application of DAG systems is still far away from reality.

Table 6: Latest Update

	Ever Existed Projects
Non-updating	Orumesh [25], Trustnode [216], DAGX [217], Soteria [137], IoT Chain [142] Intervalue [218], COTI [219]
Claim without use	Z-DAG [138], Rchain [220], NXT [221]
In-progress	3D-DAG [222][223], XDAG [224] Constellation [225] [226]

† Data resources from Coinmarketcap (https://coinmarketcap.com/), Feixiaohao (https://www.feixiaohao.com/) and Github on September 2020.

8.5 Clearing Myths.

We also collect several highly mentioned problems. At the end of paper, we provide our answers as follows.

- *Do we need blocks?* We do not always need blocks. The systems in *Type I/III/V* show us with a series of examples.
- *What's the main difference between DAG-based and classic blockchains?* The main difference is the organization of units that transits from the linear-based model to the graph-based model. Another point that is often neglected are their properties. Classic blockchain systems set strict consistency as their default requirements, whereas DAG systems may greatly weaken such requirements.
- *Does DAG indeed help to improve the performance?* DAG solutions do improve the performance of blockchain, in three aspects: scalability, throughput, and confirmation time. A system may improve one or two of them, which depends on its design of the protocol and requirements of the properties.
- *Can smart contract be supported?* Ensuring correct state transitions is the most fundamental task of smart contracts. This requires the units sorted in linear order for a consistent view across different ledgers. Thus, only the system with a linear sorting algorithm has the chance to establish an upper layer component like smart contract.
- *What types of applications can be applied?* Existing systems can only support basic functionalities like asset transferring. Upper layer components like the smart contract have not been implemented so far (except for external components such as [62]). The applications, heavily relying on state transitions, cannot be directly applied to current DAG systems.
- *Is there a blockchain design that simultaneously scales throughput, storage efficiency, and security [11]?* Currently, no protocols in the context of DAG-based blockchain systems exist, meeting all these requirements at the same time. Moreover,

the metrics mentioned in the question are even controversial among peer researchers.

- *What types of paradigm can we learn?* We have identified six types of DAG-based systems under two-dimensional metrics. No concurrent work can outpace ours with respect to the refinement of classification. Fitzi *et al.* introduces the parallel chain model, which in fact contains *Type III/IV*. Other studies [80], mentioning the blockDAG or TxDAG, cannot clearly and comprehensively present the features and differences among current DAG systems, either.

9 CONCLUSION

Constrained by limitations on performance and scalability, the revolution of classic blockchains is desperately required. DAG-based systems provide innovative models whose underlying structures enable high throughput and large scalability. However, the field has grown increasingly complex with different designs and patterns, making newcomers confused. In this work, we provide the first structured analysis of DAG-based blockchains. We provide the overview through collecting and reviewing all ever-existed and ongoing studies. Then, we abstract a general model to capture the features of DAG and identify six types of design patterns. We analyze the collected systems by respectively evaluating their structure, consensus mechanism, property, security, and performance, followed by discussions on their impacts, comparisons, and challenges. To the best of knowledge, this paper provides the first comprehensive and insightful analysis and summary of DAG-based blockchain systems, making a timely contribution to the prolific and vibrant area of this field. Optimistically, we believe new mechanisms will be progressively proposed to improve the current systems. Future developments, especially structure shift, will impact the performance, scalability, or security in a variety of ways.

REFERENCES

[1] Satoshi Nakamoto et al. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[2] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. ethereum project yellow paper.(2014), 2014.

[3] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015.

[4] Marianna Belotti, Nikola Božić, Guy Pujolle, and Stefano Secci. A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials*, 21(4):3796–3838, 2019.

[5] Muhammad Salek Ali, Massimo Vecchio, Miguel Pincheira, Koustabh Dolui, Fabio Antonelli, and Mubashir Husain Rehmani. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 21(2):1676–1717, 2018.

[6] Junfeng Xie, Helen Tang, Tao Huang, F Richard Yu, Renchao Xie, Jiang Liu, and Yunjie Liu. A survey of blockchain technology applied to smart cities: Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(3):2794–2830, 2019.

[7] Tara Salman, Maede Zolanvari, Aiman Erbad, Raj Jain, and Mohammed Samaka. Security services using blockchains: A state of the art survey. *IEEE Communications Surveys & Tutorials*, 21(1):858–880, 2018.

[8] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[9] NeonVest. The scalability trilemma in blockchain. *https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df*, 2019.

[10] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. *IACR Cryptology ePrint Archive*, 2015:1019, 2015.

[11] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. SoK: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[12] Georgia Avarikioti, Eleftherios Kokoris-Kogias, and Roger Wattenhofer. Divide and scale: Formalization of distributed ledger sharding protocols. *arXiv preprint arXiv:1910.10434*, 2019.

[13] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, 2020.

[14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72, 2014.

[15] Christopher Natoli, Jiangshan Yu, Vincent Gramoli, and Paulo Esteves-Verissimo. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. *arXiv preprint arXiv:1908.08316*, 2019.

[16] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. SoK: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.

[17] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE Access*, 8:125244–125262, 2020.

[18] Community. Bitcoin cash. *https:// www.bitcoincash.org*, 2017.

[19] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. SoK: Communication across distributed ledgers. Technical report, IACR Cryptology ePrint Archive, 2019: 1128, 2019.

[20] Juan Garay and Aggelos Kiayias. SoK: A consensus taxonomy in the blockchain era. In *Cryptographers' Track at the RSA Conference*, pages 284–318. Springer, 2020.

[21] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013(881), 2013.

[22] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[23] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value. *URL https://obyte.org/Byteball.pdf*, 2016.

[24] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[25] Huma Pervez, Muhammad Muneeb, Muhammad Usama Irfan, and Irfan Ul Haq. A comparative analysis of dag-based blockchain architectures. In *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, pages 27–34. IEEE, 2018.

[26] Georgios Birmpas, Elias Koutsoupias, Philip Lazos, and Francisco J Marmolejo-Cossío. Fairness and efficiency in dag-based cryptocurrencies. *Financial Cryptography and Data Security*, 2020.

[27] Chong Bai. State-of-the-art and future trends of blockchain based on dag structure. In *International Workshop on Structured Object-Oriented Formal Language and Method*, pages 183–196. Springer, 2018.

[28] Andrew Gorczyca and Audrey Decker. Distributed ledger witness selection in bounded width directed acyclic graphs. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 124–127. IEEE, 2019.

[29] Jiahao He, Guangju Wang, Guangyuan Zhang, and Jiheng Zhang. Consensus mechanism design based on structured directed acyclic graphs. *arXiv preprint arXiv:1901.02755*, 2019.

[30] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455, 2020.

[31] Wellington Fernandes Silvano and Roderval Marcelino. Iota tangle: A cryptocurrency to communicate internet of things data. *Future Generation Computer Systems*, 2020.

[32] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

[33] Manuel Zander, Tom Waite, and Dominik Harz. Dagsim: Simulation of dag-based distributed ledger protocols. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):118–121, 2019.

[34] Mohamed Riswan Abdul Lathif, Pezhman Nasirifard, and Hans-Arno Jacobsen. Cidds: A configurable and distributed dag-based distributed ledger simulation framework. In *Proceedings of the 19th International Middleware Conference (Posters)*, pages 7–8. ACM, 2018.

[35] Zhongli Dong, Emma Zheng, Young Choon, and Albert Y Zomaya. Dagbench: A performance evaluation framework for dag distributed ledgers. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 264–271. IEEE, 2019.

[36] Vincenzo Gabrielle Rader. Blockchain and dag, the technology which enables distributed market places.

[37] Federico Matteo Benčić and Ivana Podnar Žarko. Distributed ledger technology: Blockchain compared to directed acyclic graph. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1569–1570. IEEE, 2018.

[38] A Saad and Soo Young Park. Decentralized directed acyclic graph based dlt network. In *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, pages 158–163. ACM, 2019.

[39] Patrick Schueffel. Alternative distributed ledger technologies blockchain vs. tangle vs. hashgraph-a high-level overview and comparison. *Tangle vs. Hashgraph-A High-Level Overview and Comparison (December 15, 2017)*, 2017.

[40] Isaac Sheff, Andrew Wang, Xinwenand Myers, and Robbert Renesse. A web of blocks. *https://arxiv.org/abs/1806.06978*, 2018.

[41] Liangrong Zhao and Jiangshan Yu. Evaluating dag-based blockchains for iot. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 507–513. IEEE, 2019.

[42] Louis-Claude Canon and Emmanuel Jeannot. Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):532–546, 2009.

[43] Yixin Li, Bin Cao, Mugen Peng, Long Zhang, Lei Zhang, Daquan Feng, and Jihong Yu. Direct acyclic graph based blockchain for internet of things: Performance and security analysis. *arXiv preprint arXiv:1905.10925*, 2019.

[44] Caixiang Fan. *Performance Analysis and Design of an IoT-Friendly DAG-based Distributed Ledger System.* PhD thesis, University of Alberta, 2019.

[45] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[46] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[47] Sergio Demian Lerner. Dagcoin: a cryptocurrency without blocks. *https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf* , 2015.

[48] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds, Inc. Technical Report SWIRLDS-TR-2016*, 1, 2016.

[49] Dahlia Malkhi. Blockchain in the lens of {BFT}. 2018.

[50] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network. *https://nano.org/en/whitepaper*, 2018.

[51] Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. Scaling nakamoto consensus to thousands of transactions per second. *https://arxiv.org/pdf/1805.03870.pdf*, 2018.

[52] Quentin Bramas. The stability and the security of the tangle. 2018.

[53] B Kusmierz. The first glance at the simulation of the tangle: discrete model. *IOTA Found. WhitePaper*, pages 1–10, 2017.

[54] Vidal Attias and Quentin Bramas. Tangle analysis for iota cryptocurrency. 2018.

[55] Haibo Tian, Huizhi Lin, and Fangguo Zhang. Design a proof of stake based directed acyclic graph chain. In *International Conference on Frontiers in Cyber Security*, pages 150–165. Springer, 2020.

[56] Bartosz Kusmierz, William Sanders, Andreas Penzkofer, Angelo Capossele, and Alon Gal. Properties of the tangle for uniform random and random walk tip selection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 228–236. IEEE, 2019.

[57] Ethan Heilman, Neha Narula, Garrett Tanzer, James Lovejoy, Michael Colavita, Madars Virza, and Tadge Dryja. Cryptanalysis of curl-p and other attacks on the iota cryptocurrency. *IACR Cryptology ePrint Archive*, 2019:344, 2019.

[58] Serguei Popov. On the timestamps in the tangle. 2017.

[59] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the winternitz one-time signature scheme. In *International Conference on Cryptology in Africa*, pages 363–378. Springer, 2011.

[60] Darya Melnyk and Roger Wattenhofer. The append memory model: Why blockdags excel blockchains. 2020.

[61] Bartosz Kusmierz and Alon Gal. Probability of being left behind and probability of becoming permanent tip in the tangle v0. 2, 2018.

[62] IOTA Foundation Coordicide Team. The coordicide. *https://files.iota.org/papers/20200120_Coordicide_WP.pdf*, 2019.

[63] Serguei Popov. Iota: Feeless and free. *IEEE Blockchain Technical Briefs*, 2019.

[64] Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 729–744, 2018.

[65] Laurence Tennant. Improving the anonymity of the iota cryptocurrency, 2017.

[66] Peter Ince, Joseph K Liu, and Peng Zhang. Adding confidential transactions to cryptocurrency iota with bulletproofs. In *International Conference on Network and System Security*, pages 32–45. Springer, 2018.

[67] Umair Sarfraz, Masoom Alam, Sherali Zeadally, and Abid Khan. Privacy aware iota ledger: Decentralized mixing and unlinkable iota transactions. *Computer Networks*, 148:361–372, 2019.

[68] Sehrish Shafeeq, Sherali Zeadally, Masoom Alam, and Abid Khan. Curbing address reuse in the iota distributed ledger: A cuckoo-filter-based approach. *IEEE Transactions on Engineering Management*, 2019.

[69] A Cullen, P Ferraro, C King, and R Shorten. On the resilience of dag-based distributed ledgers in iot applications. *IEEE Internet of Things Journal*, 2020.

[70] D Cai. A parasite chain attack in iota. B.S. thesis, University of Twente, 2019.

[71] Philip Staupe. Quasi-analytic parasite chain absorption probabilities in the tangle, 2017.

[72] Gerard De Roode, Ikram Ullah, and Paul JM Havinga. How to break iota heart by replaying? In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2018.

[73] Andreas Penzkofer, Bartosz Kusmierz, Angelo Capossele, William Sanders, and Olivia Saa. Parasite chain detection in the iota protocol. *arXiv preprint arXiv:2004.13409*, 2020.

[74] Quaintance Monica and Martino and Will. Chainweb protocol security calculations. *https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_26d87f20cf8548d2927e28152babf533.pdf* , 2018.

[75] Dmitry Tanana. Avalanche blockchain protocol for distributed computing security. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–3. IEEE, 2019.

[76] Pawel Szalachowski, Daniël Reijsbergen, Ivan Homoliak, and Siwei Sun. Strongchain: Transparent and collaborative proof-of-work consensus. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 819–836, 2019.

[77] Nihat Yuva and İsmail Kırbaş. Directed acyclic graph based on crypto currency application example: Iota. In *Proceedings of the International Conference on Data Science and Applications, Porto, Portugal*, pages 26–28, 2018.

[78] Chitra1 Tarun, Quaintance Monica, Haber Stuart, and Martino Will. Agent-based simulations of blockchain protocols illustrated via kadena's chainweb. *https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_3b2d0c58179d4edd9df6df4d55d61dda.pdf* , 2018.

[79] Xavier Boyen, Christopher Carr, and Thomas Haines. Graphchain: A blockchain-free scalable decentralised ledger. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, pages 21–33, 2018.

[80] Yonatan Sompolinsky and Aviv Zohar. Phantom, ghostdag, 2020.

[81] Iddo Bentov, Pavel Hubácek, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptology ePrint Archive*, 2017:300, 2017.

[82] Shuyang Tang, Qingzhao Zhang, Zhengfeng Gao, Jilai Zheng, , and Dawu Gu. Bracing a transaction dag with a backbone chain. *https://eprint.iacr.org/2020/472.pdf*, 2020.

[83] Matthias Fitzi, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. *IACR Cryptology ePrint Archive*, 2018:1119, 2018.

[84] Jia Kan, Shangzhe Chen, and Xin Huang. Improve blockchain performance using graph data structure and parallel mining. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pages 173–178. IEEE, 2018.

[85] Marco Manoppo. Designing a better blockchain consensus.

[86] Gewu Bu, Önder Gürcan, and Maria Potop-Butucaru. G-iota: Fair and confidence aware tangle. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 644–649. IEEE, 2019.

[87] Gewu Bu, Wassim Hana, and Maria Potop-Butucaru. Metamorphic iota. *https://arxiv.org/abs/1907.03628*, 2019.

[88] Pietro Ferraro, Christopher King, and Robert Shorten. Iota-based directed acyclic graphs without orphans. *arXiv preprint arXiv:1901.07302*, 2018.

[89] Andrea Tesei, Luca Di Mauro, Mariano Falcitelli, Sandro Noto, and Paolo Pagano. Iota-vpki: A dlt-based and resource efficient vehicular public key infrastructure. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, 2018.

[90] Odysseas Lamtzidis, Dennis Pettas, and John Gialelis. A novel combination of distributed ledger technologies on internet of things: Use case on precision agriculture. *Applied System Innovation*, 2(3):30, 2019.

[91] Caixiang Fan, Hamzeh Khazaei, Yuxiang Chen, and Petr Musilek. Towards a scalable dag-based distributed ledger for smart communities. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 177–182. IEEE, 2019.

[92] IOTA Foundation. Qubic: Accessed on june 01, 2020. *https://qubic.iota.org/*, 2020.

[93] IOTA Foundation. Masked authenticated messaging module. *https://github.com/iotaledger/MAM*.

[94] Xiaochen Zheng, Shengjing Sun, Raghava Rao Mukkamala, Ravi Vatrapu, and Joaquín Ordieres-Meré. Accelerating health data sharing: A solution based on the internet of things and distributed ledger technologies. *Journal of medical Internet research*, 21(6):e13583, 2019.

[95] James Brogan, Immanuel Baskaran, and Navin Ramachandran. Authenticating health activity data using distributed ledger technologies. *Computational and Structural Biotechnology Journal*, 16:257–266, 2018.

[96] Tariq Alsboui, Yongrui Qin, and Richard Hill. Enabling distributed intelligence in the internet of things using the iota tangle architecture. 2019.

[97] Laizhong Cui, Shu Yang, Ziteng Chen, Yi Pan, Mingwei Xu, and Ke Xu. An efficient and compacted dag-based blockchain protocol for industrial internet

of things. *IEEE Transactions on Industrial Informatics*, 2019.
[98] Gerard Ruiz. Distributed data management in internet of things networking environments: Iota tangle and bitcoin blockchain distributed ledger technologies, 2018.
[99] Paulo C Bartolomeu, Emanuel Vieira, and Joaquim Ferreira. Iota feasibility and perspectives for enabling vehicular applications. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2018.
[100] Dragos Strugar, Rasheed Hussain, Manuel Mazzara, Victor Rivera, Joo Young Lee, and Ruslan Mustafin. On m2m micropayments: a case study of electric autonomous vehicles. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1697–1700. IEEE, 2018.
[101] Wenhui Yang, Xiaohai Dai, Jiang Xiao, and Hai Jin. Ldv: A lightweight dag-based blockchain for vehicular social networks. *IEEE Transactions on Vehicular Technology*, 69(6):5749–5759, 2020.
[102] Jianxiong Guo, Xingjian Ding, and Weili Wu. A double auction for charging scheduling among vehicles using dag-blockchains. *arXiv preprint arXiv:2010.01436*, 2020.
[103] Natasa Zivic, Christoph Ruland, and Jochen Sassmannshausen. Distributed ledger technologies for m2m communications. In *2019 International Conference on Information Networking (ICOIN)*, pages 301–306. IEEE, 2019.
[104] Dario Assante and Roberto Crucco. M2m learning environment for electric applications. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 1518–1522. IEEE, 2019.
[105] Alexander Raschendorfer, Benjamin Mörzinger, Eric Steinberger, Patrick Pelzmann, Ralf Oswald, Manuel Stadler, and Friedrich Bleicher. On iota as a potential enabler for an m2m economy in manufacturing. *Procedia CIRP*, 79:379–384, 2019.
[106] Jordan Murkin. A block-free distributed ledger for p2p energy trading: Case with iota? In *Advanced Information Systems Engineering*, page 111. Springer.
[107] Shengjing Sun, Xiaochen Zheng, Javier Villalba-Díez, and Joaquín Ordieres-Meré. Indoor air-quality data-monitoring system: Long-term monitoring benefits. *Sensors*, 19(19):4157, 2019.
[108] Mirko Zichichi, Stefano Ferretti, and Gabriele D'Angelo. A distributed ledger based infrastructure for smart transportation system and social good. *arXiv preprint arXiv:1910.03280*, 2019.
[109] Odysseas Lamtzidis and John Gialelis. An iota based distributed sensor node system. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2018.
[110] Chao Wu, Liyi Zhou, Chulin Xie, Yuhang Zheng, and Jiawei Yu. Data quality transaction on different distributed ledger technologies. In *International Conference on Big Scientific Data Management*, pages 301–318. Springer, 2018.
[111] Xiaochen Zheng, Alba Vieira, Sergio Labrador Marcos, Yolanda Aladro, and Joaquín Ordieres-Meré. Activity-aware essential tremor evaluation using deep learning method based on acceleration data. *Parkinsonism & related disorders*, 58:17–22, 2019.
[112] Sehrish Shafeeq, Masoom Alam, and Abid Khan. Privacy aware decentralized access control system. *Future Generation Computer Systems*, 101:420–433, 2019.
[113] Seongjoon Park and Hwangnam Kim. Dag-based distributed ledger for low-latency smart grid network. *Energies*, 12(18):3570, 2019.
[114] Andrew Cullen, Pietro Ferraro, Christopher King, and Robert Shorten. Distributed ledger technology for smart mobility: Variable delay models. *arXiv preprint arXiv:1903.12466*, 2019.
[115] Yepeng Ding and Hiroyuki Sato. Dagbase: A decentralized database platform using dag-based consensus. In *2020 IEEE 44rd Annual Computer Software and Applications Conference (COMPSAC). IEEE*, 2020.
[116] Gautam Srivastava, Ashutosh Dhar Dwivedi, and Rajani Singh. Crypto-democracy: A decentralized voting scheme using blockchain technology. In *ICETE (2)*, pages 674–679, 2018.
[117] Gautam Srivastava, Ashutosh Dhar Dwivedi, and Rajani Singh. Phantom protocol as the new crypto-democracy. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 499–509. Springer, 2018.
[118] Jamileh Bahri and Hamid Reza Shayegh Borojeni. Electronic voting through de-pbft consensus and dag data structure. In *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 391–396. IEEE, 2019.
[119] Zhiyi Zhang, Vishrant Vasavada, Xinyu Ma, and Lixia Zhang. Dledger: An iot-friendly private distributed ledger system based on dag. *arXiv preprint arXiv:1902.09031*, 2019.
[120] Pietro Ferraro, C King, and Robert Shorten. Distributed ledger technology for smart cities, the sharing economy, and social compliance. *IEEE Access*, 6:62728–62746, 2018.
[121] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.
[122] Sang-Min Choi, Jiho Park, Quan Hoang Nguyen, Andre Cronje, Ki-Young Jang, Hyunjoon Cheon, Yo-Sub Han, and Byung-Ik Ahn. Opera: Reasoning about continuous common knowledge in asynchronous distributed systems. *ArXiv*, abs/1810.02186, 2018.
[123] Sang-Min Choi, Jiho Park, Quan Nguyen, and Andre Cronje. Fantom: A scalable framework for asynchronous distributed systems. *arXiv preprint arXiv:1810.10360*, 2018.
[124] Quan Nguyen and Andre Cronje. Onlay: Online layering for scalable asynchronous bft system. *ArXiv*, abs/1905.04867, 2019.
[125] Quan Nguyen, Andre Cronje, Michael Gangyu Kong, Antonia Kampa, and George Samman. Stakedag: Stake-based consensus for scalable trustless systems. *ArXiv*, abs/1907.03655, 2019.
[126] Quan Thanh Nguyen, Andre Cronje, Michael Gangyu Kong, Alex Kampa, and George Samman. Stairdag: Cross-dag validation for scalable bft consensus. *ArXiv*, abs/1908.11810, 2019.
[127] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. *IACR Cryptology ePrint Archive*, 2018:104, 2018.
[128] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre : Serialization of proof-of-work events: Confirming transactions via recursive elections. 2017.
[129] Adam Gagol and Michal Swietek. Aleph: A leaderless, asynchronous, byzantine fault tolerant consensus protocol. *arXiv preprint arXiv:1810.05256*, 2018.
[130] Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability. *arXiv preprint arXiv:1906.08936*, 2019.
[131] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019.
[132] George Danezis and David Hrycyszyn. Blockmania: from block dags to consensus. *https://arxiv.org/pdf/1809.01620.pdf*, 2018.
[133] Lei Yang, Vivek Bagaria, Gerui Wang, Mohammad Alizadeh, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Scaling bitcoin by 10,000 x. *arXiv preprint arXiv:1909.11261*, 2019.
[134] Tai-Yuan Chen, Wei-Ning Huang, Po-Chun Kuo, Hao Chung, and Tzu-Wei Chao. Dexon: A highly scalable, decentralized dag-based consensus algorithm. *arXiv preprint arXiv:1811.07525*, 2018.
[135] Xavier Boyen, Christopher Carr, and Thomas Haines. Blockchain-free cryptocurrencies: A framework for truly decentralised fast transactions. *Cryptology ePrint Archive*, 2016.
[136] Zhaoming Yin, Anbang Ruan, Ming Wei, Huafeng Li, Kai Yuan, Junqing Wang, Yahui Wang, Ming Ni, and Andrew Martin. Streamnet: A dag system with streaming graph computing. *arXiv preprint arXiv:1908.06405*, 2019.
[137] Foundation. Soteria-dag. *https://github.com/soteria-dag/soterd/blob/master/docs/README.md*.
[138] Foundation. Z-dag withepaper. *https://syscoin.org/zdag_syscoin_whitepaper.pdf*.
[139] Chunming Liu, Daniel Wang, and Ming Wu. Vite: A high performance asynchronous decentralized application platform. *https://github.com/vitelabs/whitepaper/blob/master/vite_en.pdf*, 2018.
[140] Chenxing Li, Fan Long, and Yang Guang. Ghast: Breaking confirmation delay barrier in nakamoto consensus via adaptive weighted blocks. *arXiv:2006.01072*, 2020.
[141] A decentralized blockchain with high throughput and fast confirmation. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, July 2020.
[142] Foundation. Iot chain withepaper. *https://iotchain.io/pdf/ITCWHITEPAPER.pdf*.
[143] Zsolt István, Alessandro Sorniotti, and Marko Vukolić. Streamchain: Do blockchains need blocks? In *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, pages 1–6. ACM, 2018.
[144] Serguei Popov. Local modifiers in the tangle. *Work in progress*, 2018.
[145] Haifeng Yu, Ivica Nikolic, Ruomu Hou, and Prateek Saxena. Ohie: blockchain scaling made simple. *41st IEEE Symposium on Security and Privacy*, 2020.
[146] Jianyu Niu. Eunomia: A permissionless parallel chain protocol based on logical clock. *arXiv preprint arXiv:1908.07567*, 2019.
[147] Martino Will, Quaintance Monica, and Popejoy Stuart. Chainweb: A proof-of-work parallel-chain architecture for massive throughput. *https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_029c9991469e4565a7c334dd716345f4.pdf*, 2018.
[148] Himanshu Gupta and Dharanipragada Janakiram. Cdag: A serialized blockdag for permissioned blockchain. *arXiv preprint arXiv:1910.08547*, 2019.
[149] Pierre Chevalier, Bartlomiej Kaminski, Fraser Hutchison, Qi Ma, Spandan Sharma, Andreas Fackler, and William J Buchanan. Protocol for asynchronous, reliable, secure and efficient consensus (parsec) version 2.0. *arXiv preprint arXiv:1907.11445*, 2019.
[150] Kaituo Cao, Fei Lin, Chaohui Qian, and Keyu Li. A high efficiency network using dag and consensus in blockchain. In *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/-SocialCom/SustainCom)*, pages 279–285. IEEE, 2019.
[151] Chun-Xuan Zhou, Qiang-Sheng Hua, and Hai Jin. Hotdag: Hybrid consensus via sharding in the permissionless model. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 807–821. Springer, 2020.

[152] Zhujun Zhang, Dali Zhu, and Baoxin Mi. C-dag: Community-assisted dag mechanism with high throughput and eventual consistency. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 113–121. Springer, 2020.

[153] Marcin Abram, David Galindo, Daniel Honerkamp, Jonathan Ward, and Jin-Mann Wong. Democratising blockchain: A minimal agency consensus model. *arXiv preprint arXiv:2006.05390*, 2020.

[154] B Swaroopa Reddy and GVV Sharma. Scalable consensus protocols for pow based blockchain and blockdag. *arXiv e-prints*, pages arXiv–2010, 2020.

[155] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. In *International Conference on Cryptology and Information Security in Latin America*, pages 327–351. Springer, 2017.

[156] Lyudmila Kovalchuk, Dmytro Kaidalov, Andrii Nastenko, Oleksiy Shevtsov, Mariia Rodinko, and Roman Oliynykov. Number of confirmation blocks for bitcoin and ghost consensus protocols on networks with delayed message delivery. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 42–47. ACM, 2018.

[157] Serguei Popov, Olivia Saa, and Paulo Finardi. Equilibria in the tangle. *Computers & Industrial Engineering*, 136:160–172, 2019.

[158] Bartosz Kusmierz, Philip Staupe, and Alon Gal. Extracting tangle properties in continuous time via large-scale simulations. Technical report, working paper, 2018.

[159] Jing Li and Dongning Guo. On analysis of the bitcoin and prism backbone protocols. *arXiv preprint arXiv:1907.05016*, 2019.

[160] Niclas Kannengießer, Sebastian Lins, Tobias Dehling, and Ali Sunyaev. Mind the gap: trade-offs between distributed ledger technology characteristics. *arXiv preprint arXiv:1906.00861*, 2019.

[161] Bozhi Wang, Qin Wang, Shiping Chen, and Yang Xiang. Security analysis on tangle-based blockchain through simulation. In *Australasian Conference on Information Security and Privacy*, pages 653–663. Springer, 2020.

[162] M Divya and Nagaveni B Biradar. Iota-next generation block chain. *International journal of engineering and computer science*, 7(04):23823–23826, 2018.

[163] Gerui Wang, Shuo Wang, Vivek Bagaria, David Tse, and Pramod Viswanath. Prism removes consensus bottleneck for smart contracts. *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2020.

[164] Team Rocket. Demo visualization: Snowball protocol. *https://tedyin.com/archive/snow-bft-demo/#/snow*, 2019.

[165] Penzkofer IAndreas, Kusmierz1 Bartosz, Capossele Angelo, Sanders William, and Saa Olivia. Parasite chain detection in the iota protocol. *https://arxiv.org/pdf/2004.13409.pdf*.

[166] Luigi Vigneri, Wolfgang Welz, Alon Gal, and Vassil Dimitrov. Achieving fairness in the tangle through an adaptive rate control algorithm. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 146–148. IEEE, 2019.

[167] Adam Gagol, Damian Lesniak, Damian Straszak, and Michal Swietek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 214–228, 2019.

[168] Andrew Cullen, Pietro Ferraro, William Sanders, Luigi Vigneri, and Robert Shorten. On congestion control for distributed ledgers in adversarial iot networks. *arXiv preprint arXiv:2005.07778*, 2020.

[169] Shuai Xiao, Xu An Wang, and Han Wang. Large-scale electronic voting based on conflux consensus mechanism. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 291–299. Springer, 2019.

[170] Tong Zhou, Xiaofeng Li, and He Zhao. Dlattice: A permission-less blockchain based on dpos-ba-dag consensus for data tokenization. *IEEE Access*, 7:39273–39287, 2019.

[171] Fu Xiang, Wang Huaimin, Shi Peichang, Ouyang Xue, and Zhang Xunhui. Jointgraph: A dag-based efficient consensus algorithm for consortium blockchains. *Software: Practice and Experience*, 2019.

[172] Runchao Han, Zhimei Sui, Jiangshan Yu, Joseph Liu, and Shiping Chen. Fact and fiction: Challenging the honest majority assumption of permissionless blockchains.

[173] Vincent Gramoli. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 2017.

[174] Imran Makhdoom, Mehran Abolhasan, Haider Abbas, and Wei Ni. Blockchain's adoption in iot: The challenges, and a way forward. *Journal of Network and Computer Applications*, 125:251–279, 2019.

[175] Xu Wang, Xuan Zha, Wei Ni, Ren Ping Liu, Y Jay Guo, Xinxin Niu, and Kangfeng Zheng. Survey on blockchain for internet of things. *Computer Communications*, 136:10–29, 2019.

[176] Marko Vukolić. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 3–7. ACM, 2017.

[177] Nicola Atzei, Massimo Bartoletti, Stefano Lande, and Roberto Zunino. A formal model of bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 541–560. Springer, 2018.

[178] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[179] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Caper: a cross-application permissioned blockchain. *Proceedings of the VLDB Endowment*, 12(11):1385–1398, 2019.

[180] Sébastien Forestier, Damir Vodenicarevic, and Adrien Laversanne-Finot. Blockclique: scaling blockchains through transaction sharding in a multithreaded block graph. *arXiv preprint arXiv:1803.09029*, 2018.

[181] IOTA Foundation. Coordinator-part-1: The-path-to-coordicide. *https://blog.iota.org/coordinator-part-1-the-path\protect\discretionary{\char\hyphenchar\font}{}{}to-coordicide-ee4148a8db08*.

[182] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies, 2018.

[183] Leemon Baird, Mance Harmon, and Paul Madsen. Hedera: A governing council & public hashgraph network. *The trust layer of the internet, whitepaper*, 1, 2018.

[184] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[185] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2017.

[186] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, pages 173–186, 1999.

[187] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[188] Christian Cachin and Marko Vukolić. Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017.

[189] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

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

[191] Ling Ren. Analysis of nakamoto consensus. Technical report, Cryptology ePrint Archive, Report 2019/943.(2019). https://eprint. iacr. org …, 2019.

[192] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, pages 1–10, 2016.

[193] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer, 2015.

[194] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[195] Christopher Natoli and Vincent Gramoli. The balance attack against proof-of-work blockchains: The r3 testbed as an example. *arXiv preprint arXiv:1612.09426*, 2016.

[196] Buterin Vitalik. The problem of censorship. *https://blog.ethereum.org/2015/06/06/the-problem-of-censorship/*, 2015.

[197] Rebstock Joseph. Replay attacks in iota. 2018.

[198] IOTA Foundation. Structure of a bundle. *https://docs.iota.org/docs/iota-basics/0.1/references/structure-of-a-bundle*.

[199] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[200] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30, 2016.

[201] Tezos. Tezos github. *https://gitlab.com/tezos/tezos*, 2017.

[202] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1, 2012.

[203] Maxim Jourenko, Kanta Kurazumi, Mario Larangeira, and Keisuke Tanaka. SoK: A taxonomy for layer-2 scalability related protocols for cryptocurrencies. *IACR Cryptol. ePrint Arch.*, 2019:352, 2019.

[204] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*, pages 45–59, 2016.

[205] Jiangshan Yu, David Kozhaya, Jeremie Decouchant, and Paulo Jorge Esteves Veríssimo. Repucoin: Your reputation is your power. *IEEE Trans. Computers*, 68(8):1225–1237, 2019.

[206] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th {usenix} security symposium ({usenix} security 16)*, pages 279–296, 2016.

[207] Guillaume Vizier and Vincent Gramoli. Comchain: Bridging the gap between public and consortium blockchains. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1469–1474. IEEE, 2018.
[208] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2018.
[209] Christopher Copeland and Hongxia Zhong. Tangaroa: a byzantine fault tolerant raft, 2016.
[210] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, 2016.
[211] Sigwit. *https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki*.
[212] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
[213] Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.
[214] Juan A Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast pki setup. In *IACR International Workshop on Public Key Cryptography*, pages 465–495. Springer, 2018.
[215] Purathani Praitheeshan, Lei Pan, Jiangshan Yu, Joseph Liu, and Robin Doss. Security analysis methods on ethereum smart contract vulnerabilities: a survey. *arXiv preprint arXiv:1908.08605*, 2019.
[216] Foundation. Trustnode chain withepaper. *https://coinmarketcap.com/currencies/trustnote/*.
[217] Foundation. Dagx withepaper. *https://www.feixiaohao.com/currencies/dagx/*.
[218] INVE Official. Intervalue whitepaper. *https://www.inve.one/file/InterValue_whitepaper_en.pdf*.
[219] Coti: a decentralized, high performance cryptocurrency ecosystem optimized for creating digital payment networks and stable coins. In *https://coti.io/files/COTI-technical-whitepaper.pdf*, 2018.
[220] Foundation. Rchain website. *https://www.rchain.coop*.
[221] Foundation. Nxt website. *https://www.jelurida.com/nxt*.
[222] Qin Wang. Improving the scalability of blockchain through dag. In *Proceedings of the 20th International Middleware Conference Doctoral Symposium*, pages 34–35, 2019.
[223] Joe Zou, Zhongli Dong, Allen Shao, Peng Zhuang, Wei Li, and Albert Y Zomaya. 3d-dag: A high performance dag network with eventual consistency and finality. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pages 262–263. IEEE, 2018.
[224] XDAG Foundation. Xdag. *https://github.com/XDagger/xdag/wiki*.
[225] Foundation. Constellation website. *https://constellationnetwork.io*.
[226] Foundation. Perlin network. *https://github.com/perlin-network/*.