DAG-BLOCK: A Novel Architecture for Scaling Blockchain-Enabled Cryptocurrencies

Naina Qi, Yong Yuan, and Fei-Yue Wang, *Fellow, IEEE*

***Abstract*— With the rapid development of blockchain technology and industries, scalability has been widely realized as one of the primary and urgent concerns for the large-scale adoption of blockchain, especially for cryptocurrencies. In this respect, directed acyclic graph (DAG) proves to be an elegant solution to scaling blockchain but suffers from weak consistency and security issues. In this article, we designed a novel DAG-BLOCK architecture for blockchain-enabled cryptocurrency markets in order to improve the scalability. In our work, DAG is used to replace the Merkel-tree-based transaction structure within the block, and a novel design of open blocks is proposed to enable user nodes to participate in verifying the transactions in blockchain systems. On this basis, we designed a new segmented market structure, in which each miner serves only a group of users instead of all users, so as to reduce miners' workload and thus scale transaction processing capabilities. Our work can help improve the scalability of cryptocurrencies via evolving the underlying blockchain systems to graph-based distributed ledgers and is expected to shed new light on designing blockchain-based decentralized markets.**

***Index Terms*— Blockchain, consensus, directed acyclic graph (DAG), scalability.**

I. Introduction

BLOCKCHAIN has attracted intensive interests in recent years in both academia and industries since Satoshi Nakamoto's well-publicized Bitcoin whitepaper [1]. Technically speaking, blockchain can be used to evolve the server-based centralized architecture to serverless, consensus-based distributed ledger systems for enhanced trust, reliability, usability, and also social efficiency. In blockchain systems, all peer nodes (also known as, miners) simultaneously participate in the transmission and validation of data transactions and maintain an identical data ledger with global consistency. This decentralized design can help solve the long-standing issues including single-point of failure and data tampering, in that the data integrity will not be effected in case a minority of peer nodes are corrupted or attempt to hack the system via tampering with the ledgers. As such, blockchain can serve as a trusted, reliable, usable, and effective platform for handling financial transactions, leading to a multibillion dollar market of cryptocurrencies.

Specially, consensus can be considered as the key for large numbers of blockchain peers to reach agreements on transaction data with the presence of Byzantine malicious nodes. As a core component in most blockchain architectures, consensus algorithms play an important role in maintaining the performance of the entire blockchain system. However, in the practice of cryptocurrency mining, it is widely witnessed that major consensus algorithms, such as proof-of-work (PoW), suffer from lowered efficiency and in turn widespread criticism on its high energy consumption [2]. Therefore, in order to solve this issue, there is an urgent need to improve the scalability of the underlying blockchain architecture for more efficient cryptocurrency mining. In our work, scalability refers to the ability to process transactions in response to changes of the transaction volume and the number of participants in the blockchain network. The network is scalable if it is capable to grow adaptively along with the demand of the user base [3]. Scalability can be regarded as one of the primary and urgent concerns for blockchain system, especially cryptocurrencies. Generally speaking, throughput and latency are two major performance metrics for scalability. Taking Bitcoin as an example, although huge amount of miner computing power is invested in the hash operations to generate new blocks, the processing speed is limited to 3–7 transactions per second (TPS) at present, far below the speed of centralized financial payment platforms including VISA and Alipay that are 2 and 500 thousands TPS, respectively. Meanwhile, there are severe latency in Bitcoin in that transactions must wait for one or more consensus rounds before they can be written into blocks, and in practice, six blocks should be created and linked on-chain before transactions can be confirmed secured. Hence, current cryptocurrencies are not particularly suitable to micropayment and high-frequency trading scenarios, limiting the potential applications in the future programmable societies.

In literature, many researchers have worked on consensus algorithms to solve the scalability issue of cryptocurrencies. Eyal et al. [4] proposed Bitcoin-NG and introduced the

Manuscript received 30 June 2022; revised 3 October 2022; accepted 18 November 2022. Date of publication 13 December 2022; date of current version 31 January 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101401; in part by the National Natural Science Foundation of China under Grant 72171230; in part by the Science and Technology Development Fund, Macau, SAR, under Grant 0050/2020/A1; and in part by Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing. *(Corresponding author: Yong Yuan.)*

Naina Qi is with the School of Mathematics, Renmin University of China, Beijing 100872, China.

Yong Yuan is with the School of Mathematics, Renmin University of China, Beijing 100872, China, and also with the Engineering Research Center of Finance Computation and Digital Engineering, Ministry of Education, Beijing 100872, China (e-mail: yong.yuan@ruc.edu.cn).

Fei-Yue Wang is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the Institute of Systems Engineering, Macau University of Science and Technology, Macau, China.

Digital Object Identifier 10.1109/TCSS.2022.3224764

concept of microblocks, which generates one key block and multiple microblocks per time slot, thus can be able to create more blocks without changing the block size and help scale Bitcoin. Luu et al. [5] proposed Elastico and introduced the idea of sharding, which isolated the network into multiple shards in a provably secure manner with shards processing in parallel disjoint sets of transactions, aiming at enhancing the scalability of blockchain. Kokoros-Kogias et al. [6] put forward the Omniledger to improve blockchain scalability by processing transaction across shards in parallel. Wang [7] proposed monoxide to use asynchronous consensus zones to offer the ability of linear scaling. Each zone is responsible for its own data and thus splitting the workload of all key components without sacrificing the decentralization and security of the system. Manuskin [8] proposed Ostraka that can parallelize the nodes in fractions and demonstrated that replacing a unified node with an Ostraka node will not undermine the security of underlying consensus mechanism. Other similar solutions include Zilliqa [9], Harmony [10], and Ethereum Sharding2.0 [11], among others. These research efforts significantly increase the scalability due to the parallel executions of state shards, so that most of the scaling solutions to PoW algorithms are shard-based [12]. Meanwhile, there are many other probability-based scaling solutions. For instance, Gilad et al. [13] proposed Algorand using a randomized algorithm, which can scale up to 500 000 users with high throughput. Kiayias et al. [14] proposed Ouroborous that uses a coin flip protocol in leader election to improve the throughput and number of users. To sum up, these works can help solve the blockchain scalability issue but mainly from a perspective of innovation on PoW or Byzantine fault tolerant (BFT) consensus algorithms. To the best of our knowledge, scalability solutions based on novel blockchain architectures are still far from sufficient.

In literature, researchers have proposed to use directed acyclic graph (DAG) [15], [16] to evolve the chain-based, linear data structure underlying blockchain to the graph-based distributed ledger structure. This novel DAG network has significant advantages including enhanced efficiency and concurrence, lower power consumption, as well as asynchronous operations and parallel transaction processing capabilities. However, it suffers from a prominent risk of double spending and unverified security on large-scale systems. With the continuous expansion of DAG, the confirmation time of the transactions is not controllable. In DAG-based systems, a previous transaction is approved by later transactions, so that the tip transaction might not be approved at all time. Meanwhile, DAG systems adopt a gossip-type propagation protocol and operate asynchronously and cannot maintain strong consistency. Also, there is no global sorting mechanism, so that in case when running a smart contract, it is likely that data stored in nodes will differ after a period of time. To sum up, DAG and blockchain can complement each other, and thus, DAG-based blockchains and cryptocurrencies have attracted intensive research interests.

As shown in Fig. 1, DAG-based blockchain or distributed ledger takes a single transaction (instead of a block) as a unit in the graph. The transaction details include sender and receiver

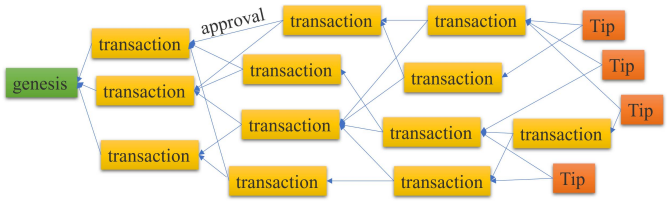

Fig. 1. Structure of a DAG.

addresses, the value transferred, transaction fee, timestamp, digital signature, and so on. The edges in a DAG denote the unidirectional hash pointers linking from the subunit to its parent unit between transactions. These hash points cannot be tampered with, so that the positions of transactions denoting their submitting order are fixed in the DAG. The process of establishing hash pointers is called "approve," and those units that have not been approved in the DAG are called "tips." In case a new transaction is added, two tips will be selected to approve, and the new transaction will become a new tip. The first transaction is also known as the Genesis transaction [17].

In this article, we aim at combining DAG and blockchain to address the scalability issues in cryptocurrencies. Our main idea is to replace the Merkel-tree-based transaction structure within the block by DAG, instead of replacing the entire chain as other solutions in literature. The advantages of a DAG-based transaction structure within blocks are not only to retain a globally linear chain to facilitate the judgment of transaction orders but also to limit the size of the DAG to avoid the issues, such as infinite DAG expansion and uncontrollable tips confirmation time. At the same time, we also proposed the concept of "open blocks," in which user nodes can independently choose tip transactions to approve, instead of having to rely on miner nodes to handle their transactions. In addition, we propose to segment the cryptocurrency ecosystems into niche markets, in which each miner serves to process transactions for only a group of users rather than in traditional peer-to-peer (P2P) networks, where all miners are responsible to process transactions for all users. This market segmentation can effectively reduce the workload of miners and thus help scale to more users and improve the speed of processing transactions.

The remainder of this article is organized as follows. In Section II, we propose the architecture of underlying market networks, the DAG-BLOCK blockchain structure, and also the detailed design of the consensus process. In Section III, we discuss several security issues including double spending, dust trading, tip selection, malicious attacks from adversaries, as well as economic incentives. Section IV discusses the key differences between our proposed DAG-BLOCK and other DAG-based cryptocurrencies including IoTA [18]. Section IV concludes and discusses our future works.

II. DAG-BLOCK Architecture

In this section, we will present the detailed design of our DAG-BLOCK architecture, including the underlying market network, the block structure, and the consensus process.

TABLE I
LIST OF NOTATIONS

Notation	Description	Notation	Description
N	Number of miner nodes	π	Proof of the user
S_i	Set of user nodes that the i-th miner node serves	$seed_k$	Seed in round k
G_i	Coinbase transaction of the i-th sub-DAG	sk_i	Secret key of miner i and sk_{k-1} is secret key of L_{k-1}
m, M	Transaction	k	Number of rounds
p	Probability of selecting 6 out of N miner nodes	$hashlen$	Length of hash output
B_k	Block in round k	w_i	weight of the miner node i
μ	User node	z	Output of the loop algorithm in leader selection
D_i^k	Sub-DAG of miner node i in round k, and D^k is the aggregated DAG in round k	w_D	Weight of the transaction D calculating by $\lg[(value\ of\ D)+1]$, and W_D is the cumulative weight of transaction D
L_k	Leader in round k, and $(L_k^1, \ldots, L_k^l)$ are l potential leaders selected in round k	$B_k^1, \ldots, B_k^l$	Block generated by leader $(L_k^1, \ldots, L_k^l)$ in round k
t_1	Time spent on packaging D_i^k	t_2	Time spent on building and linking B_k

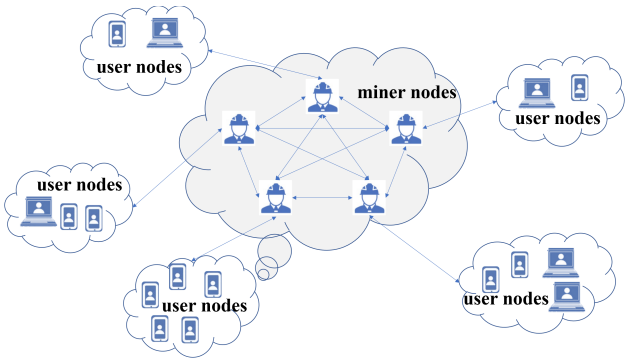

Fig. 2. Architecture of the market network.

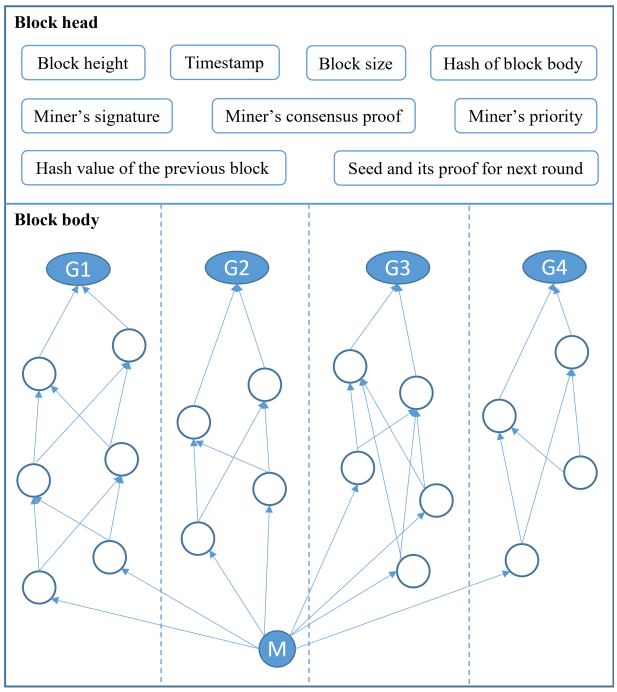

Fig. 3. DAG-BLOCK structure.

A. Architecture of Market Networks

In our design, the market network is a combination of a decentralized P2P network and centralized star networks. Similar as in Bitcoin systems, there are also two types of nodes in this network, i.e., user nodes and miner nodes.

The market network is illustrated in Fig. 2. Assume that there are N miner nodes in the P2P network, each connected with a group of user nodes with network addresses. Note that each miner node has complete transaction data of the blockchain and is responsible for serving its connected user nodes (instead of all users as in Bitcoin). We use S_i $(i = 1, 2, \ldots, N)$ to denote the set of user nodes that the ith miner node serves. For simplicity, we assume that each miner node serves a disjoint set of user nodes, i.e., $S_i \cap S_j = \emptyset$, $\forall i \neq j, i, j = 1, 2, \ldots, N$. Without loss of generality, a user node might have multiple network addresses served by different miners. For instance, a user node might have two addresses in S_i and S_j $(i \neq j)$, respectively. A miner node must store the transactions submitted by its served user nodes in the memory pool and generate a sub-DAG (i.e., a part of the complete DAG in the block) for the current consensus round. User nodes also have memory pools to store its historical transactions but do not have to store all transaction data. The relationship of the nodes is shown in Fig. 2, with a fully decentralized P2P network among miner nodes and a star network between a miner node and its served user nodes. The notation and description of all variables used in this article are listed in Table I.

B. DAG-BLOCK Structure

As shown in Fig. 3, a DAG-based block consists of a block head and a block body.

The block body encapsulates all packaged transactions using a DAG structure. Each transaction uses a hash value as its unique ID, and transactions link to each other via hash pointers. Each transaction is denoted by a circle in the DAG in Fig. 3 and consists of five elements, i.e., $m =$ [pre_hash(1), pre_hash(2), transaction, timestamp, signature(μ)], where pre_hash(1) and pre_hash(2) are two parent transactions

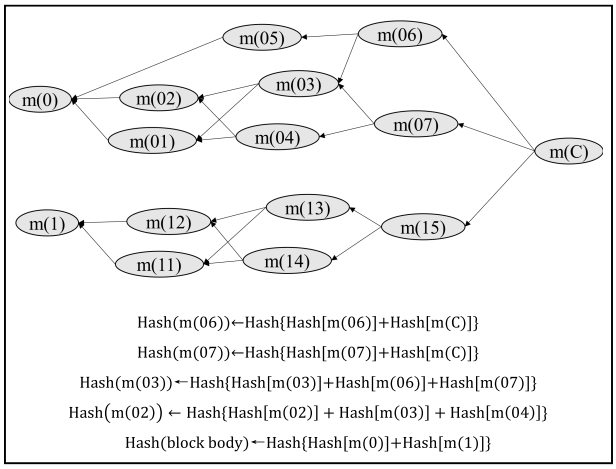

Fig. 4. Hashing the block body: an illustrating example.

approved by m, and the other three elements represent the detailed data of the transaction, user node's timestamp, and signature of the submitting user μ, respectively [19]. It is worth noting that each transaction must approve two parent transactions, except the one that approves the unique genesis transaction G_i $(i = 1, 2, \ldots, N)$ and the transaction "M". We also generate a global tip transaction denoted by "M" in Fig. 3 for each block to aggregate all miners' submitted sub-DAGs via approving their tip transactions. The generation method of transaction M will be explained in detail in Section II.

The detailed design of our DAG-BLOCK architecture is shown in Fig. 3. Typically, a block header consists of nine items including the block height, timestamp, block size, hash of the block body, miner's signature, miner's consensus proof, miner's priority, hash value of the previous block, as well as seed and its proof for next round. Here, the miner's consensus proof is the output of the verifiable random functions (VRFs) in the random leader selection process, serving as a proof of the legitimacy for the selected miner node.

The hash value of the block body is computed as the aggregated hash value of all transactions in the DAG. In case when computing the hash of block body, we must recompute the hash of each transaction m, which is the hash value of all transactions approving m and the original hash value of m. As such, we can compute the hash value of each transaction in a reverse chronological order of timestamps, from later transactions to earlier ones, until the genesis transaction. In Fig. 4, we show an illustrating example of computing the hash of block body with two miner nodes and 15 transactions. The new block is linked to the previous block using a hash pointer to guarantee the tamper-evidence and the order of the transactions.

The other detailed definitions of the above items are described in Section II.

C. Consensus Process

We assume a synchronous network in our design, with no more than 1/3 of all miner nodes is Byzantine malicious. The consensus process is divided into rounds of almost equal time length, so we use B_k to denote the block built in round $k = 1, 2, \ldots,$. In round k, the consensus process is as follows.

Step 1): All user nodes submit transactions. A user node $\mu(\mu \in S_i)$ submits a transaction m to the miner node i, where m = [pre_hash(1), pre_hash(2), transaction, timestamp, signature (μ)]. Here, pre_hash(1) and pre_hash(2) denote the parent transactions approved by m, and "transaction" is the details of the transaction m. More specifically:

a) μ *requests the local sub-DAG from its serving miner node* i. We call miner nodes' local DAG as sub-DAG, which will be used to construct the complete DAG in block body.

b) μ *receives the sub-DAG.* To reduce the complexity in communication, miner nodes can only send the information of tips to user nodes. If μ wishes to obtain the current sub-DAG in its miner node, it can attach an additional information in transaction m to miner i, so that miner i can send it to μ.

c) μ *selects and approves two tips from the tips in the sub-DAG.* Here, an algorithm can be designed to recommend tips for the user nodes, e.g., via ranking approval fees of tips.

d) μ *sends transaction m back to the miner node* i. In this step, there might be the case that two user nodes choose the same tips possibly due to the delay of network, but this does not influence the consensus due to DAG' feature of toleranting forks [20].

Step 2): All miner nodes generate the sub-DAG. Miner i keeps updating its local sub-DAG D_i^k based on user-submitted transactions and checking for conflicting or double-spending transactions submitted by user nodes according to the tip selection algorithm.

Step 3): Preparation of leader selection. According to the default duration of the round k, after a certain time interval, miner i who is the leader of round $k-1$ sends the signal $<$ *begin next stage*, *timestamp* $Publickey_i >$ to other miners j in this system, here $j = 1, 2, \ldots, N$ and $j \neq i$. Once the signal was received by miner node j, it will verify the signal, i.e., whether the time interval is correct or not, and if yes, then the state of the network moves on to the next step 4.

Step 4): Randomly select the leader and build block B_k. Randomly select l miners $\{L_k^1, L_k^2, \ldots, L_k^l\}$ to pack a global DAG D^k and build block B_k using VRF. Here, we call $\{L_k^1, L_k^2, \ldots, L_k^l\}$ potential leaders. In particular, l is not fixed and can vary with k.

Step 5): Broadcast the block. $\{L_k^1, L_k^2, \ldots, L_k^l\}$ broadcast their built blocks $\{B_k^1, B_k^1, \ldots, B_k^l\}$ as well as their own priority and proof. According to the priority, all miners will choose a block from the miner with the maximum priority and verify this block. If malicious transactions are included in the block, then all miners will choose and verify the block from the miner with the second maximum priority, until finding a valid block from a miner with lower priority. The selected miner becomes the leader of the round k, denoted by L_k, and then, other miners accept block B_k built by L_k.

Step 6): Reach agreement on a block. If B_k is legal, then all miners accept B_k and then link B_k to B_{k-1} using a hash pointer. In this situation, all honest miners will reach a consensus, i.e., they all accept an honest block with the same priority that does not include malicious transactions. Otherwise, an agreement will be reached on an empty block.

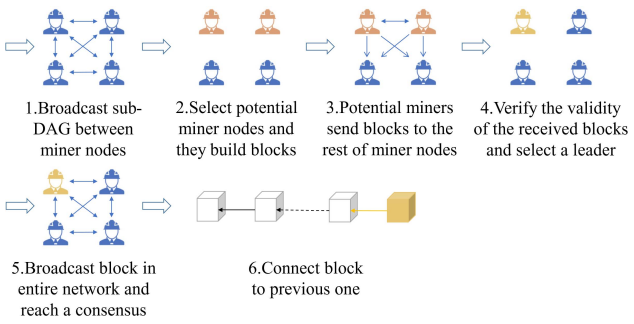

Fig. 5. Communications in the consensus process.

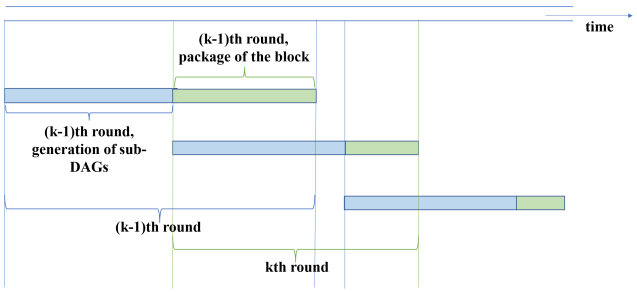

Fig. 6. Timeline of the consensus process.

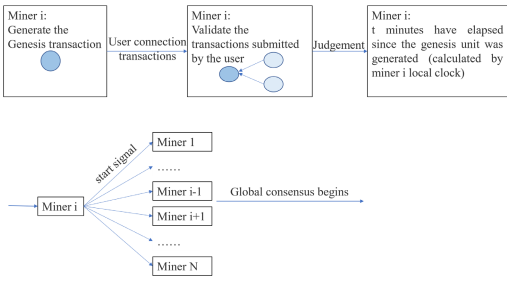

Fig. 7. Starting consensus in each round (where miner i is the leader of the previous round).

Specially, Fig. 5 illustrates the process from Step 4) to Step 6), i.e., after locally packaging sub-DAGs and verifying the signal of state transition. We focus on the communications between miner nodes here. All miner nodes first broadcast their sub-DAGs to other nodes. Then, each miner node will execute a cryptographic lottery protocol to determine whether itself is a potential leader or not when it received all sub-DAGs from others. If it is a potential leader, it will build a block. After building a block, the potential leader will transmit to the rest of all miner nodes, who will verify blocks from potential leaders and their priority and proof and choose the one with the highest priority and valid block as the unique leader in this round. Finally, all miner nodes connect the block generated by the leader to previous one.

According to the network synchronization assumption, all message will be received within a constant time period, and thus, all miner nodes will know the number of all online miner nodes in the entire network in this round and will receive all blocks broadcasted by potential leaders. Moreover, potential leaders can also decrease their communication costs via only broadcasting the block header they built. If a block header is accepted by the entire network, then its submitting miner also knows this message and will send the complete block to entire network. In this way, consensus can be reached on this block.

D. Division of Consensus Rounds

The consensus process is divided into rounds by time, and one block will be built in each round [21]. As shown in Fig. 6, each round consists of two steps, i.e., the package of D_i^k, which takes time t_1, and building and linking of B_k, which takes time t_2, where $t_1 + t_2 = t$. D_i^k is the sub-DAG packed by the miner i ($i = 1, 2, \ldots, N$) in round k using the transaction in its memory pool and will eventually be packed into blockB_k. Technically, the kth round opens when the transaction DAG D_i^k starts forming and closes when B_k is finally appended onto the blockchain. Because it takes time for potential leader miners to pack blocks, there might be overlaps between rounds, in case when $t_1 > t_2$ obviously.

As shown in Fig. 7, when the leader miner node starts building B_k, i.e., packing all sub-DAGs of round k into a candidate block, it will start forming the sub-DAG of round $k + 1$ at the same time, and after time t_1 in this leader's local clock (note that there is no global clock in our decentralized framework), the leader miner node sends a signal of starting consensus of round $k + 1$ to all other miner nodes. Then, the global consensus starts after other miners verify that the time interval is t_1 [22], approximately. The leader initiates the signal for starting the consensus that also ensures $t_1 > t_2$.

E. Formation of D_i^k

As shown in Fig. 6, when the transaction DAG formed in round $k - 1$ is packed into block, miner node j (if j is not the leader of the $k - 1$ round) will move to round k, and j will have a "public sub-DAG," where j is responsible for publishing Genesis transaction.

1) When a user node μ belonging to S_j is willing to connect a new transaction, miner j will send a real-time updated public sub-DAG to μ it served, and the new transaction will be broadcast through the star network, so that the user node has a real-time updated DAG for round k. However, the user node does not have to store all the data of the transaction DAG in round k and thus achieving a consistent state synchronization of the distributed ledger containing the new transaction.
2) A user node μ submits and connects transactions in this public sub-DAG. The user node μ sends its transaction to miner node j, who can in turn choose the tips to be approved according to the transaction information. Hash pointers are used to link the transactions, and two tips are selected as parent transactions of the new transaction. A tip selection protocol can be designed in the framework for recommending tips for μ without the complete data of the system.
3) Miner j verifies whether there is a conflict between the new transaction and the historical transactions that its direct or indirect approval and completes the approval

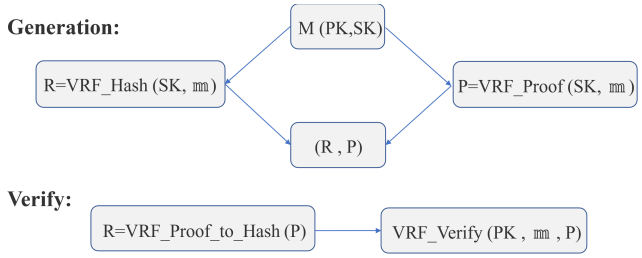

Fig. 8. Process of VRF.

of the DAG tips, so as to realize the confirmation of the new transaction in the sub-DAG.

4) Miner j is responsible for verifying whether the transactions from its served user nodes are malicious or not. In case when there is a conflict with the DAG ledger, the transaction will be rejected by miner j. Meanwhile, miner j completes the anti-fraud test of whether or not the new transaction is double spending. If there are two conflicting transactions, the miner node can discard one of them to maintain the security and consistent of this system as every transaction in sub-DAG is not confirmed at this stage.

F. Random Leader Selection

Inspired by Algorand, a cryptography lottery protocol is designed in our work using VRF, and a miner node is selected randomly in each round for building a block. The protocol is used to create a random number generator to determine the building miner (i.e., leader) for the round. Each miner node is weighted and the probability of being selected as the leader is proportional to the number of transactions and the total number of balances of user nodes it serves.

1) VRF's Operational Mechanisms: In each round, the election will take a seed as an input to the VRF, which will then output a random hash (as seed input for the next round) and π (the proof of the user selected by lottery within this round). The seed here is also the hash input value, and the seed for the genesis block will be generated using a distributed random number generator. All other miner nodes can use public key to verify the correctness of the output.

The operation process of VRF is shown in Fig. 8, where m is the original input message. The VRF contains a key generation algorithm that generates a public VRF key (PK) and a secrete VRF key (SK). Verification is conducted by using the pair of public keys to verify that P is a proof generated based on the original message m.

2) Cryptographic Lottery Protocol:

1) Calculating $seed_k$. If there is an elected miner in round $k-1$, then

$$< seed_k, \pi > \leftarrow VRF_{sk_{k-1}}(seed_{k-1}||k).$$

Here, the signature of the elected miner in round $k-1$ for the $seed_{k-1}$ is first used, and $seed_{k-1}$ is then calculated using VRF, which produces a proof and $seed_k$. Otherwise, $seed_k = H(seed_{k-1}||k)$ when the block is an empty block in round $k-1$, in which all potential leaders' blocks are malicious with invalid transactions or there is no agreement on a valid block in round $k-1$. $Seed_0$ is generated by distributed random number generator in the beginning of the consensus process.

2) Determining the potential leader. When $seed_k$ is broadcast to all miners, each miner can determine whether it is a potential leader or not. For a miner node i ($i = 1, 2, \ldots, N$), it has a hash and a proof π through

$$VRF_{sk_i}(seed_k) \rightarrow < hash, \pi >$$

where sk_i is the secrete key of miner i, and the hash is within $[0, 2^{\text{hashlen}} - 1]$. We here assume $p = (6/N)$, where 6 is the excepted number of potential leaders that has been proven reasonable in the Algorand system. In extreme cases, there is no selected potential leader, so we can design a minimum guarantee to ensure the existence of a potential miner, so as to ensure that the honest leader in the previous round is always the potential leader in the next round. Based on the above assumption, every miner node starts a loop as follows.

Starting the loop from $z = 0$, $z \in \{0, 1, \ldots, w_i\}$

When

$$\frac{\text{hash}}{2^{\text{hashlen}}} \notin \left[\sum_{r=0}^{z} B(r; w_i, p), \sum_{r=0}^{z+1} B(r; w_i, p)\right]$$

$$z = z + 1.$$

When the loop is finished, the algorithm will return

$$< hash, \pi, z >$$

where w_i is the weight of the miner node i, and

$$B(r; w, p) = \binom{w}{r} p^r (1-p)^{w-r}, \quad \sum_{r=0}^{w} B(r; w, p) = 1.$$

The binomial distribution ensures that a malicious node will not increase its advantage via sybil attacks since

$$B(r_1 + r_2; w_1 + w_2, p) = B(r_1; w_1, p) + B(r_2; w_2, p).$$

Finally, this algorithm will return a parameter z, which indicates how many times the miner node i is selected. If $z > 0$, the miner node i will be selected as a potential leader.

Assume that there are l potential leaders $\{L_k^1, \ldots, L_k^l\}$. Of course, the fewer is the potential leaders, and the fewer communications will be required. In order to prevent sybil attacks, there is a canonical calculation model that divides the users in each round equally, based on the number of tokens in the wallet and the number of transactions initiated in the previous round, so that each user node represents several "average user nodes." Assume that there are N miner nodes in total, w_i is the number of "average user nodes" that miner i serves, i is also the user node miner i serves, and thus, $W = \sum_{i=1}^{N} w_i$, where W is the total weight of the system.

3) Generating blocks by potential leaders. If a node believes itself to be a potential leader, it starts the process of building a block.

4) Broadcasting blocks. Potential leaders $\{L_k^1, L_k^2, \ldots, L_k^l\}$ send blocks being packed in to others.

5) Identifying the leader node. The one with the largest z among all potential leaders is identified as the leader. As such, the parameter z can be viewed as the priority of the

miner node. The leader's block is selected for verification and confirmed in the entail network, and a network-wide consensus will be reached on the new block. If the miner with the largest z is malicious, all miners will reach consensus on the block from the miner with the second maximum priority and so on, or the honest miner will reach an agreement to treat this block as an empty block. After a specified time interval, the previous leader of this round will send the signal of starting the next round in this situation.

G. Building Blocks

After time t_1 since the formation of the genesis transaction in D_i^k, miner nodes will stop the transaction approval for D_i^k, and the sub-DAG in each miner's memory pool stops the new transaction approval. Then, miner i who is the leader of round $k-1$ sends D_i^k and a signal of starting consensus at this point to all remaining miners. After other miners verify that this signal is legitimate such as time interval being within a specified range, then they all send their own sub-DAG of the transactions to all remaining miners. Then, all miners verify the received transactions in D_j^k $(j = 1, 2, \ldots, N)$. Meanwhile, some potential miners are randomly selected to build a block. The potential miners produce a coinbase transaction "M" as the last tip unit of the total DAG, linking each miner's tip transactions to the coinbase transaction via a hash pointer (such as depicted in Fig. 3, where $N = 4$), i.e., all tip transactions are approved by a coinbase transaction. The coinbase transaction includes hash value of all tips of the sub-DAGs generated in round k and other related information, such as the publishment of transaction fee and reward of the miner nodes. In this manner, the total transaction DAG D^k of the entire network in round k can be established. When potential leaders generate blocks $\{B_k^1, B_k^1, \ldots, B_k^l\}$ and send them to all miners for verification based on priority, once verified, B_k will be linked to B_{k-1}.

III. Analysis of Security Issues

Security is a vital component for most decentralized cryptocurrencies. In this section, we discuss several classic attack scenarios, including double spending attacks, dust transaction attacks, and other malicious attacks. We also discuss the related issues of tip selection and economic incentives, which play a key role for the economic stability and security.

A. Double Spending Transactions

To avoid double spending, a transaction can only be finally confirmed if it is packed into a block and appended onto the blockchain.

If two double spending transactions are located in different blocks, then it is relatively easy to determine which one should be confirmed according to the block order. Based on the strict setting of round interval, all miner nodes know that this round will build a unique block whether it is valid or empty, then it is hard for malicious miner nodes to withhold blocks in case to replace the honest chain as all honest miner nodes will not accept simultaneously two blocks and more. Meanwhile, all miner nodes will reach an agreement on a unique priority based on the network synchronization assumption as well as the consensus on priority in the block header.

Otherwise, if they are located in the same block, we adopt the IoTA mechanism to discard double spending transactions. However, it is not difficult based on the setting that each user node address can only be added to one miner's serving set, so the double spending transactions will not be located in two sub-DAGs from two different miners. In this case, all miner nodes can discard randomly one of the conflicted transactions when they find them in packing sub-DAGs. In addition, if the miner node did not find the malicious transactions and the malicious transactions will be located in the same block, it is relatively simple by just using the IoTA mechanism to determine which transaction is effective. It is worth noting that in our design, the global DAG in each block can be appropriately sized so as to facilitate the determination of double spending transactions. Similarly, if double spending transactions are generated in memory pools, then one of the transactions can be randomly abolished, since the transaction in memory pools is not confirmed. Also, all transactions in memory pools will be quickly connected to the global transaction DAG.

B. Dust Transactions

As in most cryptocurrency systems, large numbers of tiny but bulky transactions flooding the network can cause congestion of the network, and attackers can conduct parasitic chain attacks by posting tiny transactions. For the security of the network and to avoid parasitic chain attacks, dust transactions must be prevented from broadcasting to the network, and miner nodes receiving dust transactions must not approve them to a sub-DAG.

In Bitcoin practice, if the transaction fee is higher than 1/3 of the transaction value, it might be considered as a dust transaction. Generally speaking, the minimum volume of a pay-to-public key hash (P2PKH) transaction in Bitcoin with one input and one output totals $148 + 34 = 182$ bytes, and the transaction fee is determined by the minRelayTxFee (0.00001BTC/KB by default) preconfigured for each node, so the transaction fee is "$182/1000 * \text{minRelayTxFee}$." So, the transaction value should not be lower than 0.00000546 BTC. Otherwise, it will be identified as a dust transaction.

Inspired by Bitcoin, we explicitly consider dust transactions in our work in order to avoid parasitic chain attacks and other malicious behaviors. First, the minimum amount of the transaction can be estimated depending on its type. In case when its output is below a predefined threshold, the transaction will not be constructed. Particularly, if this output happens to occur as a change, dust transactions can be avoided by discarding that part of the dusty output to act as a transaction fee.

C. Malicious Attacks From Adversaries

There might be malicious attacks, and here, we discuss several potential scenarios.

First, in case when the selected miner with the highest priority is malicious, we offer two possible strategies of malicious

miners and two solutions to deal with those cases. One strategy is to send the same malicious block to all honest miners. If the block is empty or if the block contains a malicious transaction, then the honest miner nodes will discard the block and reach an agreement on an empty block. In addition, as the DAG of transactions of all miners has been broadcast network widely before the block is broadcast, the construction method of the total DAG is basically determined, because our work is designed with the role of potential leaders that send the blocks along with a proof of priority to the miners, so the honest miners discard the block and choose the next block in order of priority until the honest block is chosen. Therefore, the method described above is valid and effective to determine the malicious block. Also, the second potential strategy is to send honest blocks to some of the honest miners and malicious blocks to the remaining honest miners. In this case, when the number of malicious miner nodes does not exceed 1/3 and the block is verified and broadcast by all miner nodes, the malicious block will not be committed and all honest miner nodes will select an empty block, and then, consensus is reached.

Second, in case when a miner node submits a sub-DAG containing a malicious transaction or containing a double spending transaction, an honest miner will not accept the entire sub-DAG as the miner node should have verified the malicious transaction before submitting the sub-DAG for consensus. If the miner realized that its sub-DAG is not going to be packed into the block in this round, it will verify the sub-DAG again, reconnect transactions in this sub-DAG, and wait to be packaged into the next block. There will not be a situation where an honest sub-DAG is not packed.

Meanwhile, in addition to setting up incentive mechanisms, we can use penalty mechanisms to maintain the market ecosystem. Each miner node address is associated with a wallet that must deposit a certain amount of digital currency. The behavior of the miner is monitored by the entire network, i.e., all miner nodes. If a miner node or user node discovers the malicious behavior from a miner node, it can record the evidence and package it into a transaction, then connect it to the sub-DAG it is responsible for, and submit this evidence onto the blockchain together with the sub-DAG composed of transactions. After the consensus process, the miner will be punished for his malicious behavior by decreasing the corresponding amount from his wallet, distributing it to the users who discovered the malicious behavior and electing a new miner responsible for this part of the users. The malicious miner will no longer be selected as a miner node, and transformation of data or transaction verification will not be performed by the remaining miners.

D. Selection of Tips

When a user node submits a transaction, it will request the latest sub-DAG from its corresponding miner node. In order to reduce the communication cost, the miner nodes can rank the transactions to offer the user node recommendations and only send the necessary information of tips or other transactions to user node instead of the complete information of the sub-DAG. According to our designed incentive mechanism, each transaction has an approval fee related to its size and the amount of the transaction fee. As such, one natural recommendation algorithm offered to user nodes is based on the list of approval fees in descending order, and thus, user nodes only need to know the order of approval fee rather than the complete information of the DAG ledger.

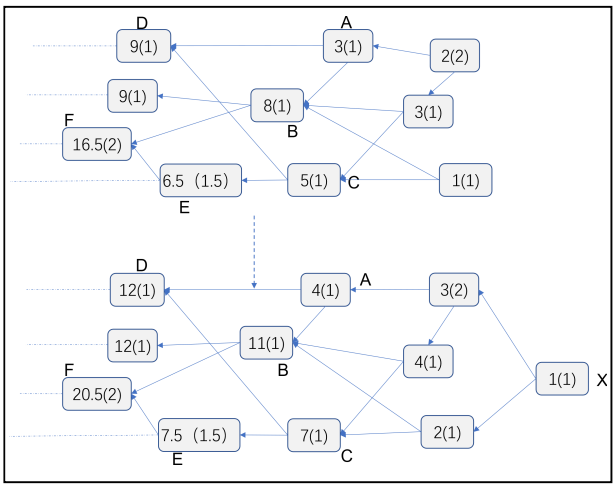

Fig. 9. Calculation of cumulative weights: an illustrating example.

The second algorithm is based on the selection algorithm of tips as in the existing DAG-based ledger (e.g., IoTA, Byteball [23], Hashgraph [24], GHOST [25], PHANTOM [26]) to avoid double spending. The sequence of timestamps mentioned earlier is used to compute the hash of the block body. The timestamps are based on local clocks in a decentralized network and thus are not theoretically reasonable in determining double spending transactions. Therefore, an algorithm is needed to sort and select the tip transactions.

Third, a weighting mechanism is introduced to help identify double spending. The value of weight is positively related to the value of the transaction. However, in order to prevent the transaction value from being too large, we use the logarithmic function to calculate the weights W, i.e., $W = lg$ (value of the transaction $+ 1$). This can also prevent transactions containing double spending with too high cumulative weight to increase the possibility of a successful double payment attack. The cumulative weight is calculated as Fig. 9.

The numbers in brackets denote the weight of the transactions. Before the transaction X was submitted to the sub-DAG, the cumulative weight of transaction D is

$$W_D = w_D + W_A + W_C = 1 + 3 + 5 = 9.$$

After X was submitted, the cumulative weight is

$$W_D = w_D + W_A + W_C = 1 + 4 + 7 = 12.$$

With this calculation, an attacker cannot massively increase the cumulative weight of a double spending transaction by exploiting the generation of a small number of transactions with a larger value.

According to this weighting mechanism, we now present the tip selection algorithm based on the Markov Chain Monte

Carlo (MCMC) to protect against parasitic chain attacks. The specific implementation of the MCMC algorithm is as follows.

1) Taking $\mathcal{R}$ as a relatively large cumulative weight value, all transactions with cumulative weights in $[\mathcal{R}, 2\mathcal{R}]$ will be called the set $\mathcal{A}$.

2) Scattering $\mathcal{N}$ walkers randomly in each transaction of the set $\mathcal{A}$.

3) Random walkers move toward the tips in the direction of the DAG edge, calculate their arrival time at the tips, discard the walkers that arrive the tips too quickly, and select the tip that the walker reaches fastest as the approved transaction on this basis.

4) A walker is most likely to move toward the transaction with the largest cumulative weight when encountering multiple bifurcated edges. If the probability of moving from any transaction x to transaction y is denoted as P_{xy}, then

$$P_{\mathcal{X}\mathcal{Y}} = \frac{\exp(-\alpha(W_{\mathcal{X}} - W_{\mathcal{Y}}))}{\sum_{\mathcal{Z}:\mathcal{Z}\to\mathcal{X}} \exp(-\alpha(W_{\mathcal{X}} - W_{\mathcal{Z}})}$$

where $\mathcal{Z}$ is the other subtransaction of $\mathcal{X}$, and $W_{\mathcal{X}}$ is the cumulative weight value of the transaction $\mathcal{X}$.

In the operation of the system, whether it is an ordinary user error or a malicious attack by adversaries, conflicts between transaction are inevitable, even if we limit the size of the DAG within blocks. In order to prevent a transaction from being replaced by conflicting transactions, our system adopts an algorithm to easily discriminate conflicting transactions, thus preventing the system from double-payment attacks. The rule of discriminating conflicting transaction algorithm is: running the tip selection algorithm based on MCMC multiple times, counting the number of times when two conflicting transactions are indirectly approved by the selected tips, and selecting the transaction with the highest number of approvals as the legal transaction.

Fourth, two conflicting transactions are not allowed to exist in the same block. If the miner node is honest, it will discard randomly one of the transactions in packing sub-DAG to safeguard the security of the system. If there are two conflicting transactions in the same block in case when the miner node is malicious, all honest miner nodes will discard this sub-DAG, which is propagating by the malicious miner node to maintain the consistence of the consensus.

E. Economic Incentives

In addition to the above efforts, designing incentive compatible economic mechanisms will help avoid malicious or strategic behaviors and ensure miner and user nodes make the expected honest actions.

As such, regarding economic incentives, there is a transaction fee submitted by the user node for each transaction, which can be divided into three parts: one for the first node to approve this transaction (so as to prevent lazy nodes from directly approving a premature transactions), i.e., the approval fee; one for the miner serving this user node, i.e., the transmission fee, and the third part to the miner who packaged the block at this stage, i.e., the block packing fee.

In our design, the transaction fee is related to the volume of transaction and the value of transaction. As the natural principle of randomly selected leaders, the reward obtained by each miner is proportional to the number of user nodes that it is serving and the value of transactions initiated by those user nodes. According to the statistic rules, reasonable assumptions can be made that the value of transactions received by every miner node follows a normal distribution, so that the reward of miner nodes is partially proportional to the mean of value of transaction. As such, theoretically, transaction fees obtained by miner nodes can be balanced. In addition, miner nodes need a higher bandwidth to transfer a larger transaction, so it is reasonable that the transaction fees are proportional to the volume of transactions. Finally, when a miner node has more user nodes, it has more chance to deal with more transactions, which is conducive to the scale of the system. To sum up, according to this method, our system is economically stable, and miner nodes will be motivated to scale the number of user nodes and maintain the benefit of user nodes, i.e., maintain the system stability.

IV. Comparison With Other DAG-Based Architectures

DAG has been widely adopted as an underlying architecture in the existing distributed ledgers and cryptocurrencies, such as IoTA, Byteball, and Hashgraph. In this section, we will discuss the major differences between our DAG-BLOCK design and other DAG-based architectures, taking IoTA as an example. To summarize, as a novel architecture, DAG-BLOCK differs with IoTA in three important dimensions, i.e., data structure, network model, and consensus algorithm.

First, for the data structure, DAG-BLOCK still maintains a chain-based linear structure in its underlying blockchain and embeds the DAG-based transaction graph into blocks. This is particularly different with the system-wide network structure in IoTA and other DAG-based architectures. This design can restrict the size of transaction DAG within each block and can help solve the long-standing issues of uncontrollable confirmation time of tip transactions caused by the infinitely expanding of DAG networks and thus has the potential of enhancing the overall consistency and security.

Second, for the network model, we adopt a two-layer network model and there are two roles in the network, i.e., the miner node and the user node. The miner nodes are responsible for proposing and verifying transactions, reaching consensus and recording all transactions, and this is the same as the IoTA nodes. However, in our DAG-BLOCK model, a miner node is allowed to serve a certain amount of user nodes, which need only submit transactions and choose tips. As such, the number of miner nodes can be far less than that of user nodes, and the required communication overhead can be reduced compared with IoTA, especially in large-scale networks. Meanwhile, we propose the open block mechanism to allow user nodes choose tips independently, which can both help reduce the workload of user nodes and also give them the chance to participate in the consensus process to improve the decentralization level.

Third, for the consensus algorithm, we propose to use random leader selection in our consensus algorithm, where a leader needs to be elected to generate new blocks in each round, and the block needs all miners' verification before

appending into the blockchain. While in IoTA blockchain, all miners can append transactions to DAG and verify them in later stages.

V. Concluding Remarks

In this article, we aim at designing a novel DAG-BLOCK architecture for blockchain-enabled cryptocurrency markets in order to improve the scalability. Technically, we propose to replace the Merkel-tree-based transactions structure within the block by DAG and also propose to use open blocks, so that user nodes can participate in verifying the transactions. On this basis, we design a new market structure in which each miner serves only a group of users, so as to reduce miners' workload and thus help scale to more users and improve the speed of processing transactions.

In our future work, we will extend our work in the following aspects. First, we plan to construct an economically incentive compatible mechanism to justify our transaction fee design based on game theoretic analysis. Second, we plan to formally prove and evaluate the consistency, security, and performance of the algorithms for our proposed algorithms. Third, we will release the synchronous network assumption and investigate our architecture in an asynchronous network model. Fourth, we plan to collect real-world data so as to empirically evaluate the scalability of our proposed architecture.

References

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Feb. 23, 2019. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] S. King. (Nov. 2014). *Ppcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[3] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–34, May 2020, doi: 10.1145/3316481.

[4] I. Eyal, A. E. Gencer, and E. G. Sirer, "Bitcoin-NG: A scalable blockchain protocol," in *Proc.13th Usenix Conf. Netw. Syst. Design Implement.*, 2016, pp. 45–59.

[5] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 17–30.

[6] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 583–598.

[7] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement.*, 2019, pp. 95–112. [Online]. Available: https://www.usenix.org/system/files/nsdi19-wang-jiaping.pdf

[8] A. Manuskin, M. Mirkin, and I. Eyal, "Ostraka: Secure blockchain scaling by node sharding," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Sep. 2020, pp. 397–406.

[9] The ZILLIQA Team. *The ZILLIQA Technical Whitepaper*. Accessed: Nov. 13, 2021. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf

[10] *Harmony*. Accessed: Nov. 26, 2021. [Online]. Available: https://harmony.one

[11] V. Buterin. (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. Accessed: Feb. 2, 2022. [Online]. Available: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

[12] J. Yadav and R. Shevkar, "Performance-based analysis of blockchain scalability metric," *Tehnički Glasnik*, vol. 15, no. 1, pp. 133–142, Mar. 2021.

[13] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. (Apr. 2018). *Algorand: Scaling Byzantine Agreements for Cryptocurrencies*. Accessed: Sep. 11, 2021. [Online]. Available: https://eprint.iacr.org/2017/454.pdf

[14] A. Kiayias, A. Russell, B. David, and R. Oliynykov. *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol*. Accessed: Feb. 15, 2022. [Online]. Available: https://eprint.iacr.org/2016/889.pdf

[15] K. Cao, F. Lin, C. Qian, and K. Li, "A high efficiency network using DAG and consensus in blockchain," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 279–285.

[16] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of DAG-based blockchain architectures," in *Proc. 12th Int. Conf. Open Source Syst. Technol. (ICOSST)*, Dec. 2018, pp. 27–34.

[17] Y. Yuan and F. Wang. *Blockchain Theories and Methods*. Beijing, China: Tsinghua Press, 2019.

[18] [Online]. Available: https://www.iota.org/

[19] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," Cryptol. ePrint Arch., Paper 2016/1159, 2016. Accessed: Sep. 18, 2021. [Online]. Available: https://eprint.iacr.org/2016/1159.pdf

[20] Y. Lewenberg, Y. Sompolinsky and A. Zohar. (2015). *Inclusive Block Chain Protocols*. Accessed: May 4, 2021. [Online]. Available: http://fc15.ifca.ai/preproceedings/paper_101.pdf

[21] E. Androulaki and C. Cachin. *Christopher Ferris, etc. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains*. [Online]. Available: http://vukolic.com/fabric.pdf

[22] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm (extended version)," in *Proc. USENIX Annu. Tech. Conf.*, Philadelphia, PA, USA, 2014, pp. 305–319.

[23] A. Churyumov. *Byteball: A Decentralized System for Storage and Transfer of Value*. Accessed: Dec. 11, 2021. [Online]. Available: https://obyte.org/Byteball.pdf

[24] L. Baird. *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*. Accessed: Dec. 11, 2021. [Online]. Available: https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[25] Y. Sompolinsky and A. Zohar. *Secure High-Rate Transaction Processing in Bitcoin (Full Version)*. Accessed: Sep. 21, 2021. [Online]. Available: https://eprint.iacr.org/2013/881.pdf

[26] Y. Sompolinsky and A. Zohar. *PHANTOM: A Scalable BlockDAG Protocol*. Accessed: Dec. 26, 2021. [Online]. Available: https://eprint.iacr.org/2018/104.pdf

Naina Qi received the B.A. degree in mathematics and applied mathematics from Shandong University, Shandong, China, in 2020. She is currently pursuing the M.S. degree in probability and mathematical statistics with the School of Mathematics, Renmin University of China, Beijing, China.

Her current research interests include blockchain and consensus algorithms.

Yong Yuan received the B.S., M.S., and Ph.D. degrees in computer software and theory from the Shandong University of Science and Technology, Shandong, China, in 2001, 2004, and 2008, respectively.

He is currently a Professor with the School of Mathematics, Renmin University of China, Beijing, China, and also with the Engineering Research Center of Finance Computation and Digital Engineering, Ministry of Education, Beijing. He has authored over 150 articles published in academic journals and conferences. His current research interests include blockchain, cryptocurrency, and smart contracts.

Dr. Yuan is also an Associate Editor of the IEEE Transactions on Computational Social Systems and ACTA Automatica SINICA. He is the Chair of the IEEE Council on RFID Technical Committee on Blockchain, the Co-Chair of the IEEE SMC Technical Committee on Blockchain, and the Director of the Chinese Association of Automation Technical Committee of Blockchain. He is the Secretary General of the IEEE SMC Technical Committee on Social Computing and Social Intelligence, the Vice Chair of the IFAC Technical Committee on Economic, Business, and Financial Systems (TC 9.1), and the Chair of the ACM Beijing Chapter on Social and Economic Computing. He is also the Secretary General of the Chinese Association of Artificial Intelligence Technical Committee on Social Computing and Social Intelligence, and the Vice Director and the Secretary General of the Chinese Academy of Management Technical Committee on Parallel Management.

Fei-Yue Wang (Fellow, IEEE) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined The University of Arizona, Tucson, AZ, USA, in 1990, and became a Professor and the Director of the Robotics and Automation Laboratory and the Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center, Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Chinese Talents Program from the State Planning Council, and in 2002, was appointed as the Director of the Key Laboratory of Complex Systems and Intelligence Science, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory for Management and Control of Complex Systems. He is currently a Professor with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, and also with the Institute of Systems Engineering, Macau University of Science and Technology, Macau, China. His current research focuses on methods and applications for parallel intelligence, social computing, and knowledge automation.

Prof. Wang is a fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and became an Outstanding Scientist of ACM for his work in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009 and 2011, respectively. In 2014, he received the IEEE SMC Society Norbert Wiener Award. Since 1997, he has been serving as the General or Program Chair of over 30 IEEE, INFORMS, IFAC, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007; the Chinese Association for Science and Technology, USA, in 2005; the American Zhu Kezhen Education Foundation from 2007 to 2008; the Vice President of the ACM China Council from 2010 to 2011; and the Vice President and the Secretary General of the Chinese Association of Automation from 2008 to 2018. He was the Founding Editor-in-Chief (EiC) of the *International Journal of Intelligent Control and Systems* from 1995 to 2000, the IEEE ITS Magazine from 2006 to 2007, the IEEE/CAA JOURNAL OF AUTOMATICA SINICA from 2014 to 2017, and the *China's Journal of Command and Control* from 2015 to 2020. He was the EiC of the *IEEE Intelligent Systems* from 2009 to 2012, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS from 2009 to 2016, and the EiC of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS since 2017, and the Founding EiC of *China's Journal of Intelligent Science and Technology* since 2019. Currently, he is the President of CAA's Supervision Council, IEEE Council on RFID, and Vice President of IEEE Systems, Man, and Cybernetics Society.