IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received January 4, 2019, accepted March 6, 2019, date of publication March 21, 2019, date of current version April 8, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2906637*

# DLattice: A Permission-Less Blockchain Based on DPoS-BA-DAG Consensus for Data Tokenization

**TONG ZHOU[1,2], XIAOFENG LI[1], AND HE ZHAO[1]**
[1]Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China
[2]University of Science and Technology of China, Hefei 230026, China

Corresponding author: He Zhao (zhaoh@hfcas.ac.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602435, in part by the Natural Science Foundation of Anhui Province under Grant 1708085QF153, and in part by the Major Project of Science and Technology of Anhui Province under Grant 16030901057.

**ABSTRACT** In today's digital information age, the conflict between the public's growing awareness of their own data protection and the data owners' inability to obtain data ownership has become increasingly prominent. The emergence of blockchain provides a new direction for data protection and data tokenization. Nonetheless, existing cryptocurrencies such as Bitcoin using Proof-of-Work are particularly energy intensive. On the other hand, classical protocols such as Byzantine agreement do not work efficiently in an open environment. Therefore, in this paper, we propose a permission-less blockchain with a novel double-DAG (directed acyclic graph) architecture called DLattice, where each account has its own Account-DAG and all accounts make up a greater Node-DAG structure. DLattice parallelizes the growth of each account's Account-DAG, each of which is not influenced by other accounts' irrelevant transactions. DLattice uses a new DPoS-BA-DAG(PANDA) protocol to reach consensus among users only when the forks are observed. Based on proposed DLattice, we introduce a process of data tokenization, including data assembling, data anchoring, and data authorization. We implement DLattice and evaluate its performance on 25 ECS virtual machines, simulating up to 500 nodes. The experimental results show that DLattice reaches a consensus in 10 seconds, achieves desired throughput, and incurs almost no penalty for scaling to more users.

**INDEX TERMS** Blockchain, data tokenization, consensus algorithm, byzantine agreement protocols, directed acyclic graph.

## I. INTRODUCTION

In this new era of digital information age, people generate a variety of data in their daily lives. On the Internet, they leave browse records and social data. In the Internet of Things, user's health data is collected by wearable devices, and usage data is acquired by smart home applications. Massive amounts of data are used to analyze behavioral and health conditions without the user's knowledge. To make matters worse, criminals use privacy data for blackmail and extortion. The US technology giant Facebook has leaked more than 50 million users' personal information data, reaping huge profits and even affecting the US election [1]. Coincidentally, China Huazhu, a large hotel group, has been reported that more than 100 million users' private data has been stolen by hackers and used for public sales online and blackmail [2]. In the scientific community, Piero Anversa, a well-known professor and leading expert in the cardiovascular field, was identified by his Harvard Medical School and Brigham and Women's Hospital as having 31 papers suspected of data fraud, and major medical journals are requested to withdraw published papers [3]. From these events we can see:

1) The control over the data is difficult to determine. Data is not controlled by the real owner (e.g. the user entity) but by the data producer (such as the equipment manufacturer or the service provider). The real owner of the data lacks the permission to agree and know about the use of data, so that the privacy is not guaranteed [4].

2) Data reliability is poor and can be falsified. Data producers have the ability to tamper with data or even fabricate false data in the centralized database, making it difficult for data collectors (e.g. research institutions), data producers, and real owners of data to establish data trust relationships.

The associate editor coordinating the review of this manuscript and approving it for publication was Feng Lin.

3) Data cannot be shared efficiently. Because the ownership of the data does not belong to the real owner, resulting in that the data cannot be shared to other paid data collectors conveniently [5].

The new blockchain technology has shown promising natures to solve these issues. Blockchain is a cryptographically secure transactional singleton machine with shared-state [6], which contains an ordered list of records linked together through chains, trees or DAGs, etc. Blocks are generated by distributed nodes through a consensus mechanism and spread and verified across the whole network [7]. The distributed nature of blockchain implies no single entity controls the ledger, but rather the participating peers together validate the authenticity of records. It is indeed because of its features such as decentralization, tamper-resistance and network data-sharing, blockchain has tremendous potential in the fields of data protection and tokenization [8], [9].

Unlike cryptocurrencies which are created on and derived their values directly from the blockchains, digital assets are often issued by real world entities and blockchains are merely a medium to record their existence and exchanges [10]. Multichain [11] offers ledgers for storing and tracking asset history. IOTA [12] issues its token and offers its public ledger as a platform for micro-payment, which makes data been exchanged among IoT devices. Previously, we proposed a method of data assetization and may help promote the data value transferring and data sharing among the Internet of Things based on Ethereum Smart Contracts [5].

Resolution of forks is the core problem faced by any cryptocurrency. Bitcoin [13] and most other cryptocurrencies [6], [14] address forks problem using Proof-of-Work (PoW), where users must repeatedly compute hashes to grow the blockchain, and the longest chain is considered authoritative [15]. The process is particularly energy intensive and time consuming. Proof-of-Stake (PoS) [16] avoids the computational overhead of proof-of-work and therefore allow reducing transaction confirmation time. On the other hand, although the advantage of original PBFT [17] is finality, which once a block is appended, it is final and cannot be replaced, Byzantine Agreement protocols do not work in an open environment efficiently because of bandwidth limitation and having no trusted public-key infrastructure [18].

The main contributions of this paper are as follows:

1) We propose a permission-less blockchain, called DLattice, with a novel Double-DAG architecture where each account has its own Account-DAG and all accounts make up a greater Node-DAG structure. DLattice parallelizes the growth of each account's Account-DAG, each of which is not influenced by other accounts' irrelevant transactions. The use of Red-Black Merkle Tree in the account's D-Tree speeds up the efficiency of querying and inserting data assets.
2) We design a new DPoS-BA-DAG (PANDA) protocol to reach consensus among users only when the forks are observed instead of executing consensus at a fixed interval. Experimental results show that the protocol can reach a consensus with latency in 10 seconds while scaling to more users.
3) We introduce a process of data tokenization based on proposed DLattice structure, including data assembling, data anchoring and data authorization.

The rest of the paper is organized as follows: Section II consists of related works, Section III reviews the preliminaries used throughout this paper. In Section IV, the blockchain model is described in detail. A series of methods for data tokenization is presented in Section V. Our PANDA consensus is elaborated in section VI, followed by the attack vectors and security analysis in Section VII. Section VIII presents the implementation and evaluation. Finally, the conclusion and future direction are presented.

## II. RELATED WORKS

### A. PROOF OF WORK (POW) VARIANTS

PoW protocols require miners to solve complex cryptographic puzzles which is easy to be verified based on their own computing power by cryptographic hashes. Specifically, the solution is a random nonce $n$ such that

$$H(n||H(b)) \leq M/D,$$

for a cryptographic hash function H with a variable number of arguments and range [0, M], a target difficulty D and the current block content $b$ [10], [19]. The faster the puzzle is solved by miners, the higher possibility a block is created. A new block is generated every 10 minutes on average in Bitcoin [20].

### B. PROOF OF STAKE (POS) VARIANTS

PoS protocols change the puzzle's difficulty to be inversely proportional to the miner's stake in the network [10], [19]. Let bal() be the function that returns the stake, then a miner $S$ can generate a new block by solving the puzzle of the following form:

$$H(n||H(b)) \leq bal(S)M/D.$$

Casper [21] is an Ethereum's upcoming PoS protocol based on smart contract. It allows miners to become validators by depositing Ethers to the Casper account. The contract then picks a validator to propose the next block according to the deposit amount. If the block is confirmed, the validator gets a small reward. But if it is not, the validator loses its deposit [10].

### C. BYZANTINE CONSENSUS VARIANTS

Byzantine Agreement (BA) protocols have been used to replicate a service across a small group of servers [22] [23], therefore they are suitable for permissioned Blockchain. PBFT [17] is deterministic and incurs $o(N^2)$ network messages for each round of agreement where N is the number of nodes in the network. Tendermint [24] proposes a small modification on top of PBFT. Instead of having an equal vote, each node in Tendermint may have different voting power, proportional to their stake in the network.

Algorand [15] uses a BA* protocol to reach consensus among users on the next set of transactions based on Verifiable Random Function that allows users to privately check whether they are selected to participate in the BA, and to include a proof of their selection in their network messages.

### D. DIRECTED ACYCLIC GRAPHS (DAGS) VARIANTS

Nano and HashGraph are some recent proposals for increasing Bitcoin's throughput by replacing the underlying chain structured ledger with a DAG structure. Hashgraph [25] is proposed for replicated state machines with guaranteed byzantine fault tolerance and achieves its fast, fair and secure transactions based on gossip about gossip and virtual voting techniques.

Nano [26] proposes a novel block-lattice architecture where each account has its own blockchain, delivering near instantaneous transaction speed and unlimited scalability and allowing them to update it asynchronously to the rest of the network, resulting in fast transactions with minimal overhead.

## III. PRELIMINARIES

### A. SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHY

Symmetric Cryptography uses the same cryptographic keys for both encryption of plaintext and decryption of cipher text. The keys may be identical or there may be a simple transformation to go between the two keys. Asymmetric Cryptography is also known as a public key cryptography and uses public and private keys to encrypt and decrypt data. The keys are simply large numbers that have been paired together but are not identical. One key in the pair can be shared with everyone and called the public key. The other key, called the private key, in the pair is kept secretly.

### B. CRYPTOGRAPHIC HASH FUNCTION

maps arbitrarily long strings to binary strings of fixed length. And it should be hard to find two different strings x and y such that H(x) = H(y), where $H()$ represents the hash function.

### C. DIGITAL SIGNATURES

allow users to authenticate information to each other without sharing any secret keys, based on public key cryptography. To create a digital signature, the hash of the message is created firstly. The private key is then used to encrypt the hash. The encrypted hash is the digital signature.

## IV. MODEL DESCRIPTION

### A. DEFINITIONS

#### 1) CONSENSUS-PARTICIPATING NODE

Consensus-participating node, $CNode_i \in \{CNode_1, .., CNode_N\}$, where N represents the number of nodes in the system. A consensus-participating node is a piece of software running on a computer that conforms system protocols and joins in the system network. The nodes communicate with each other through the gossip protocol, and their distribution is as shown in Fig. 1. The *CNode* is responsible for recording the asset ledger and the data ledger, and these ledgers in each *CNode* are the same. When initialized, each *CNode* creates one unique consensus account $ConAC_i$. The account consists of a public-private key pair $< Pk_i, Sk_i >$. The public key *Pk* is called the account address, which is used to identify the identity of $CNode_i$, and is exposed to the whole network. The consensus account needs to reserve a certain amount of consensus deposit. At the same time as enjoying the consensus right to obtain other accounts' fork penalty, the node also bears the risk of forfeiture of the consensus deposit for its malicious behaviors. Significantly, the node that owns all tokens of the system at initial state is called the Genesis Node, and it is responsible for the booting of the system.

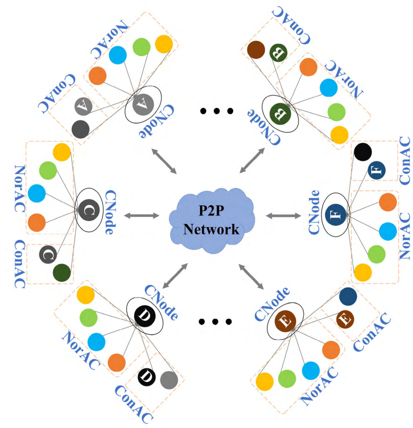



**FIGURE 1.** Overall structure of DLattice. The nodes (CNode), consisting of normal accounts (NorAC) and consensus accounts (ConAC), communicate with each other through the gossip protocol.

#### 2) ACCOUNT

The account, $Account_k \in \{Account_1, \ldots, Account_M\}$, where M represents the number of accounts, which has no theoretical upper limit. The account is the main body of actual user's participation in the system, and is composed of a public-private key pair $< Pk_k, Sk_k >$, where the public key $Pk_k$ is used to identify the identity of the account and is exposed to the entire network. A user can control multiple accounts, but each account only corresponds to one public key. The private key $Sk_k$ is similar to the password in the ordinary system. The user holding the private key has the actual control of the account. The $Sk_k$ can be used by the account to sign the transaction block or message to clarify the source of them. The accounts include normal account *NorAC* and consensus account *ConAC*. The *NorAC* consists of a currency ledger and a data ledger, which can be used to send and receive currency assets and data assets and to assign access control of the data assets. The structure of account is also shown in Fig. 1. The *ConAC* has the same function as the *NorAC* except for the functions described in Definition 1. Each account has its own DAG structure called Account-DAG, which together makes up the Node-DAG.

### 3) TRANSACTION AND BLOCK

A transaction is an agreement or a certain behavior between the sender and the receiver [27]. In this system, the construction of the transaction requires the account owner to sign with its private key, and a block contains only one transaction, so it is called a Transaction Block in this paper, recorded as *TB*. The transaction blocks include Creating Transaction Block $TB_{create}$, Delegating Transaction Block $TB_{delegate}$, Sending and Receiving Transaction Block $< TB_{send}, TB_{receive}|TB_{deal} >$, and Authorization Transaction Block $TB_{auth}$, etc., as shown in Fig. 2. Both the transfer of the currency asset and the data asset require the confirmation of two transaction blocks. The *TB* consists of three states, $\sigma(State_{TB}) \in \{S_{sending}, S_{pending}, S_{received}\}$, namely the Sending State $S_{sending}$, the Pending State $S_{pending}$ and the Received State $S_{received}$. In a transaction process, the $TB_{send}$ is constructed and broadcasted by the sender. At this time, the state of the $TB_{send}$ is $S_{sending}$. After all nodes are received (or the consensus is completed), the state becomes $S_{pending}$. When the receiver is online, the currency assets or data assets will be received according to $TB_{send}$, the corresponding $TB_{receive}$ or $TB_{deal}$ is constructed, and the $TB_{send}$'s state becomes $S_{received}$, and the entire transaction process is completed.

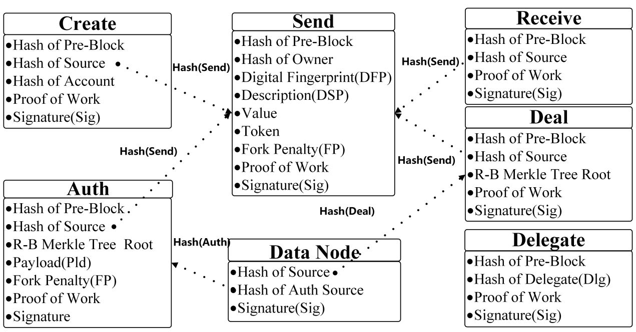



**FIGURE 2.** **Anatomy of transaction blocks.**

The Creating Transaction Block $TB_{create}$ is used to create user accounts, including creating normal accounts and consensus accounts. The initial *DLTs* of an account come from system allocation or currency transferred from other accounts. $TB_{create} = (H_{PRE}, H_{source}, H_{account}, PoW, Sig)$, where $H_{PRE}$ represents the hash value of the previous transaction block, here is the hash value of the Genesis Header; $H_{source}$ indicates the account address of sender;$H_{account}$ stands for the account address which created this transaction (the public key); *PoW* means the proof of work required to generate this transaction block; *Sig* records the signature of this transaction block with the account's private key.

Sending and Receiving Transaction Block $TB_{send}$, $TB_{receive}$ and $TB_{deal}$ are used to send or receive currency assets or data assets. The Sending Transaction Block $TB_{send} = (H_{PRE}, H_{owner}, H_{dfp}, Dsp, Value, Token, FP, PoW, Sig)$, where $H_{owner}$ represents the address of the account receiving the block; $H_{dfp}$ is a digital fingerprint of the data; *Dsp* briefly describes the sending data asset; *Value* stands for the price of the data; *Token* represents the amount of currency sent; If only *Token* and no *Value* is in $TB_{send}$, it just means the transfer of currency. The$TB_{receive} = (H_{PRE}, H_{source}, PoW, Sig)$, where $H_{source}$ indicates the hash value of the corresponding $TB_{send}$. If *Value* is included in $TB_{send}$, it indicates the transfer of data assets. The $TB_{deal} = (H_{PRE}, H_{source}, H_{RBMerkle}, Work, Sig)$, where $H_{RBMerkle}$ stores the root of D-Tree of the Account-DAG. When $< TB_{send}, TB_{receive} >$ and $< TB_{send}, TB_{deal} >$ appear in pairs, it indicates that the transfer of currency assets or data assets has been confirmed by the system. The $TB_{data}$ is the representation of $TB_{deal}$ on the D-Tree of the Account-DAG. The $TB_{data}$ is denoted as$TB_{data} = (H_{source}, H_{auth}, PoW, Sig)$, where $H_{source}$ represents the hash value of the corresponding $TB_{deal}$ and $H_{auth}$ is the hash value of the $TB_{auth}$.

Authorization Transaction Block $TB_{auth}$ is used by the account to determine which account has the access control of the data assets. The Authorization Transaction Block, $TB_{auth} = (H_{PRE}, H_{source}, H_{RBMerkle}, Pld, FP, Pow, Sig)$, where *Pld* records the list of access permission generated by the account through a mixed cryptogram arithmetic (see Section V for details). It is worth noting that these transactions are sent and received from the same account.

Delegating Transaction Block $TB_{delegate}$ is used to assign a consensus node to wield voting power on its behalf. $TB_{delegate}$ is denoted as $TB_{delegate} = (H_{PRE}, H_{DLG}, PoW, Sig)$, where $H_{DLG}$ represents the public key of the delegate node. It is worth noting that $TB_{delegate}$ only indicates that the node is delegated to wield voting power, and the actual currency assets in the account are not transferred.

### 4) DIGITAL ASSERT

Digital assets are assets in the form of electronic data which are owned or controlled by enterprises, organizations or individuals and are held for sale or in production [27]. In the proposed system, the digital assets are categorized as Currency Asset (CA) and Data Asset (DA), where CA is the token issued by the system, denoted as *DLT*, which is consumed as the equivalent in the process of data transfer and is an important part of data tokenization and a representation of data value. DA is the result of data tokenization. By assembling the raw data and storing it in a distributed database, and protecting the corresponding data fingerprint on the chain, the raw data is tokenized as on-chain assets for sale and transaction.

### 5) DLATTICE

As shown in Fig. 3(a), DLattice is a DAG structure called Node-DAG, which consists of a Genesis Header and the Account-DAG of accounts. All accounts are organized in the form of Merkle Patricia Tree (MPT) [28] by the Genesis Header. The public key of the consensus account is used as the Key, and the hash value of $TB_{create}$ which is created as an Account Root Block (ARB) is used as the Value to jointly build the MPT. The Account-DAG structure of each account is derived sequentially from its ARB, and

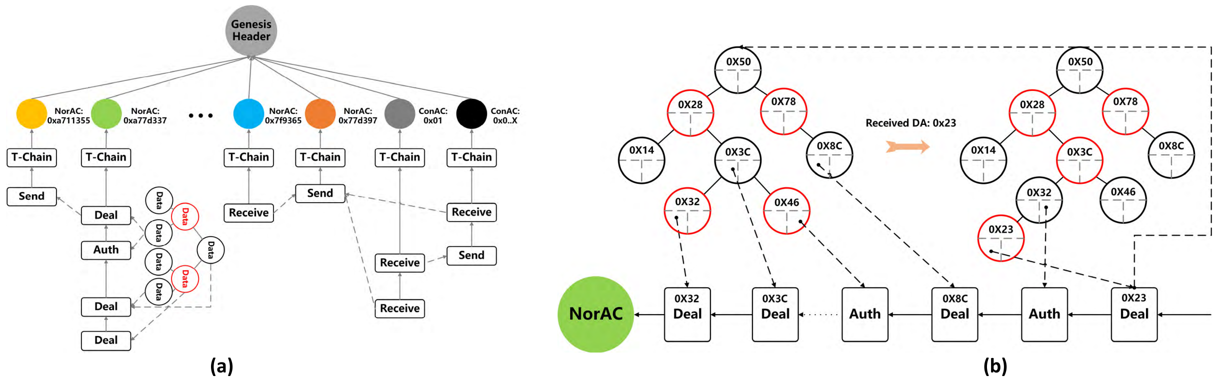



**FIGURE 3.** (a) Overall structure of Node-DAG; (b) Anatomy of Account-DAG. An older Red-Black Merkle Tree is on the left. After receiving a new DA (hash 0x23), the newer Red-Black Merkle Tree is on the right. And the $TB_{deal}$ records the Red-Black Merkle Tree root before updating.

is composed of the Token-Chain (T-Chain) and the Data-Tree (D-Tree). The income and expenditure records of the data asset and the currency asset, which are sent by the account, is recorded by T-Chain in the form of a unidirectional chain.

D-Tree is a Red-Black Merkle Tree [29] combining with T-Chain, which stores the digital fingerprint of the data asset and corresponding access control permissions, as shown in Fig. 3(b). The digital fingerprint of the data in $TB_{send}$ is taken as the Key, while $TB_{data}$ is used as the node to jointly build the Red-Black Tree. The Merkle Root of the Red-Black Tree is recorded in $H_{RBMerkle}$ of $TB_{deal}$. The complete DLattice structure is formed by the Account-DAG of all accounts.

## B. ASSUMPTIONS

*Assumption 1:* DLattice makes standard cryptographic assumptions such as public-key signatures and hash functions.

*Assumption 2:*DLattice assumes that honest users run bug-free software and the fraction of money held by honest users is above some threshold $h$(a constant greater than 2/3), but that an adversary can participate in DLattice and own some money.

*Assumption 3:* DLattice makes a "strong synchrony" assumption: most honest users can send messages that will be received by most other honest users within a known time bound $\delta_{term}$. And this assumption does not allow network partitions.

*Assumption 4:*DLattice also makes a "weak synchrony" assumption: the network can be asynchronous for a long but bounded period of time. After an asynchrony period, the network must be strongly synchronous for a reasonably long period again.

*Assumption 5:*DLattice assumes that if some probability $p$ is negligible, it means it happens with probability at most $O(1/2^{\lambda})$ for some security parameter $\lambda$. Similarly, if some event happens with high probability, it happens with probability of at least $1 - O(1/2^{\lambda})$.

## C. NOTIONS

Through this paper, we use these notions as shown in Table 1.

**TABLE 1.** Notions and detailed description.

| Notions | Meaning |
|---|---|
| $\lambda$ | Security parameters; |
| $h$ | Threshold of DLTs held by honest users in total DLT; |
| $N$ | Number of consensus nodes in the current system; |
| $\delta_{term}$ | A certain time that the messages sent by most honest users (e.g., 95%) can be received by most other honest users; |
| $\delta_{MaxTerm}$ | Maximum term in a consensus process; |
| $C_B$ | Amount of boot nodes; |
| $C_E$ | Expected size of a consensus committee; |
| $C_P$ | Actual size of a consensus committee; |
| $Committee_{seed}$ | A consensus committee consisting of consensus identities; |
| $Epoch_{Node}$ | Epoch of system development. $Epoch_{Node}$ includes the Boot Epoch $E_{Booting}$ and the Freedom Epoch $E_{Freedom}$; |
| $\tau_{good}$ | Threshold of the honest identity in the total identity, including $\tau_{good}^{boot}$ in the Boot Epoch and $\tau_{good}^{freedom}$ in the Freedom Epoch; |
| $Phase_{Consensus}$ | Phase of consensus algorithm. $Phase_{consensus}$ includes the Vote Phase $P_{vote}$ and the Commit Phase $P_{commit}$; |
| $M_{type}$ | Message types. $M_{type}$ includes Vote Messages $M_{vote}$ and Commit Messages $M_{commit}$. |

# V. DATA TOKENIZATION

## A. DATA ASSEMBLING

Data assembling is to assemble the raw data $D_{raw}$ into a data structure that can be used by DLattice at the generation source of data. This data structure is denoted as, $D = (Pk_P, Pk_O, E_K(D_{raw}), E_{EK}, T, Sig_{SK}(D_{raw}))$, where $Pk_P$ represents the public key of data producer; $Pk_O$ represents the public key of data owner; the source of $D_{raw}$ are rich and varied: it can be the continuous data generated by the device in Internet of Things, or a digital file, or a medical file, or recorded data generated between people's communication (e.g. cases, prescriptions between doctors and patients etc.). The types of $D_{raw}$ include: binary stream (images, documents, videos, etc.), URL, etc. $E_K(D_{raw})$ indicates that the data producer uses a random AES key to symmetrically encrypt the raw data, and the AES key is asymmetrically encrypted with the $Pk_O$ of data owner and is stored in $E_{EK}$. Only the data owner who owns the corresponding private key $Sk_O$ can decrypt the ciphertext to obtain $D_{raw}$. $Sig_{SK}(D_{raw})$ indicates that the data producer uses its private key $Sk_P$ to sign $D_{raw}$, the signature can be verified only by the $Pk_P$ of the data producer. $T$ represents the timestamp of the data generation. The data structure $D$ can be stored in a distributed database (IPFS [30] currently) to obtain a digital fingerprint. The digital fingerprint is stored in the $H_{dfp}$ of $TB_{send}$.

### B. DATA AUTHORIZTION

If the data owner wants to authorize the data to other accounts for access, the public key $Pk_{other}$ is needed for asymmetrical encryption of $< key, iv >$. As shown in (1), $edk_k$ is stored in the $Pld$ of the $TB_{auth}$. If the user obtains $edk_k$ from the $Pld$ of the $TB_{auth}$, and acquires the symmetric encryption key by using his private key $Sk_{other}$ and (2), the data can be decrypted by using (3), thereby realizing the access control of the data asset by using the hybrid encryption mechanism.

$$edk_k \leftarrow E_{ECC}(< key, iv > ||Pk_{other}) \tag{1}$$

$$< key, iv > \leftarrow D_{ECC}(edk_k||Sk_{other}) \tag{2}$$

$$D_{raw} \leftarrow D_{AES}(Cipher|| < key, iv >) \tag{3}$$

### C. DATA ANCHORING

Data anchoring is to anchor the digital fingerprint of data assets on the blockchain after data assets in distributed storage. Data anchoring is the core of data tokenization. The digital fingerprint obtained from data storage is used to construct a $TB_{send}$ and it will be broadcasted to the entire network. Once $TB_{send}$ is received by all the consensus nodes in the system, it will be added to the T-Chain of the corresponding account. If there is a fork, it may be appended after the consensus is reached (see Section VI for details). When the receiver is online, the data asset is checked first to see whether the demands of the receiving account are met, and then the $TB_{deal}$ (the current Red-Black Merkle Tree Root is saved in $H_{RBMerkle}$) will be created to receive the digital asset and pay. Finally, the $TB_{data}$ is created, and the Red-Black Merkle Tree is updated on its D-Tree to complete the transfer of data assets.

## VI. DPoS-BA-DAG(PANDA)

### A. NODE BOOTSTRAPPING

The development of system is divided into the Boot Epoch and the Freedom Epoch as the number of consensus nodes increases, denoted as $\sigma(Epoch_{node}) \in (E_{boot}, E_{freedom})$. In the initial Boot Epoch, the Genesis node reviews the online and storage capabilities of the new nodes (these nodes may be trusted large medical institutions, companies or government research institutions, etc., which are endorsed by their social credibility), and assigns certain initial $DLT_{init}$ to the consensus account of these nodes to complete the joining of the boot node. The committee consisting of boot nodes is called $BootCommittee$, and its size is $[4, C_B]$. The nodes less than the threshold $\tau_{good}^{boot}$ are allowed to be accidentally offline, and the $DLT_{total}$ satisfies:

$$DLT_{total} = C_B \times DLT_{init},$$

where $C_B$ represents the amount of boot nodes. In the Boot Epoch, each node knows the exact number of nodes in the current system. When the allocation of $DLT_{total}$ is completed (the amount of nodes $N > C_B$ at this moment), the system enters the Freedom Epoch, and the newly joined nodes can be added to the system at will by purchasing $DLT$ from other accounts in the system. It is noteworthy that common users can create a normal account at any time by purchasing $DLT$ from a node that has joined the system.

### B. FORK OBSERVATION

It can be seen from Definition 3 that in DLattice, the transaction block $TB$ can only be constructed by the sender, so it is impossible to be forged by a third party, which means that a malicious account can generate a fork by constructing different $TB_{send}$s with an identical previous hash $H_{PRE}$ on its own T-Chain.

Assume that an account has constructed multiple transaction blocks with identical $H_{PRE}$, as shown in Fig. 4, recorded as $List_{TB}$={$TB_{send}$, $TB'_{send}$, $TB''_{send}$, ....}, and broadcasted them to the entire network. A node will observe a set {$TB_{send}$, ..., $TB'_{send}$} with identical $H_{PRE}$, thus forming a fork. Because T-Chain is a unidirectional chain, it is necessary for all nodes in the network to pick a certain $TB$ from $List_{TB}$ and add it to its Account-DAG by the consensus algorithm.

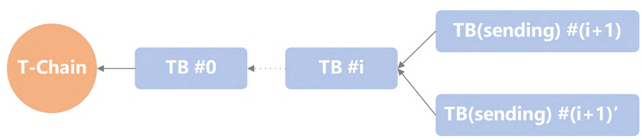



**FIGURE 4. A fork occurs when two (or more) signed transaction blocks reference the same previous block. Older transaction blocks are on the left; newer blocks are on the right.**

If a node not observe any forks, the $TB$ will be added to Account-DAG directly. When a fork is observed by a node (the node is called a Candidate Consensus Node, denoted as $Candidate_{seed}$, where $seed$ is the corresponding $H_{PRE}$), due to the incentive of Fork Penalty, the $Candidate_{seed}$ who wants to get Fork Penalty will actively participate in the consensus following these steps in Fig. 5.

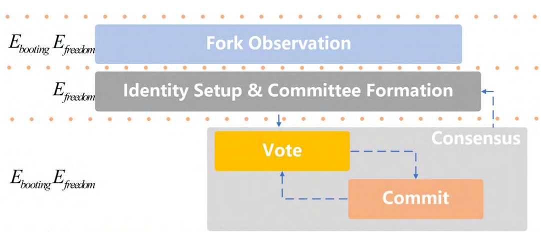



**FIGURE 5. Flow chart of PANDA consensus.**

### C. CONSENSUS IDENTITY SETUP

When a fork is observed, each $Candidate_{seed}$ begins to calculate its own consensus identities and participate in the consensus to resolve this fork.

In the $E_{boot}$, each $Candidate_{seed}$ has a unique consensus identity:

$$ID_{seed} \leftarrow < Pk, Seed, hash, proof, message, Sig_{sk} >$$

at each phase of each term of the consensus. The generation of *hash*,*proof* and *message* will be detailed in Part D.

In the $E_{freedom}$, the hash value:

$$hash \leq w_i/W$$

that satisfies the PoS condition is calculated locally and secretly by $Candidate_{seed}$ to constitutes a consensus identity:

$$ID_{seed} \leftarrow < Pk, Seed, ID_{pos} < hash >, Sig_{sk} >,$$

together with the information such as its public key to participate in this consensus. It is worth noting that nodes are able to generate multiple identities in local secretly for consensus at each phase of the consensus based on their voting power in the $E_{freedom}$. Where $w_i$ is the sum of the voting power the node holds and represents; $W$ is the total voting power. These parameters together determine the computational difficulty of the consensus identity. It can be obtained from Lemma 1 that the larger the voting power of the node, the more consensus identities will be generated in the same number of attempts, as well as the greater the probability of obtaining Fork Penalty.

### D. COMMITTEE FORMATION

The $Candidate_{seed}$ secretly generates the consensus identity in local, and the consensus committee that resolves this fork is also formed at the same time. The consensus committee is denoted as $Committee_{seed}$. In the $E_{boot}$, each $Candidate_{seed}$ generates a unique consensus identity to participate in $Committee_{seed}$. In the $E_{freedom}$, each $Candidate_{seed}$ generates multiple consensus identities secretly and locally based on its voting power to form a $Committee_{seed}$, as shown in Algorithm 1. And the $Committee_{seed}$ at the $P_{vote}$ and $P_{commit}$ are different, denoted as $Committee_{seed(vote)}$ and $Committee_{seed(commit)}$ respectively.

The ConsensusIDGeneration() (Algorithm 1) is used to generate the consensus identity at the phase $P_{consensus}$ in $e$ term. According to Lemma 2, each node calculates the consensus identity for $C_E$ times secretly and locally based on its voting power. If the identity conforms to the PoS condition, it can vote in the corresponding consensus phase.

The Verifiable Random Function (VRF) [31] is used to calculate a hash value secretly and locally, and the consensus identity that satisfies the PoS condition is calculated according to the hash value. The identity satisfies that each node can only calculate its own consensus identity instead of being calculated in advance by other nodes, while other nodes can verify the identity only after being broadcasted.

**Algorithm 1** ConsensusIDGeneration(): Generation of Consensus Identity

**Input:** $ctx, Seed, P_{consensus}$
**if** $ctx.E_{boot}$ **then**
  $< hash, proof > \leftarrow VRF_{Sk}(Seed||P_{consensus}||e)$
  $ID_{seed} \leftarrow < Pk, Seed, hash, proof, Sig_{sk} >$
  **if** $e\%\delta_{MaxTerm} == 1$ **then**
    $ctx.List_{ID}[Seed][e][P_{consensus}].append(ID_{seed})$
**else if** $ctx.E_{freedom}$
  **for** $index = 0; index < ctx.C_E; index++$
    $< hash, proof > \leftarrow VRF_{Sk}(Seed||P_{consensus}||e||index)$
    $message \leftarrow < Seed, P_{consensus}, e, index >$
    $ID_{pos} \leftarrow < hash, proof, message >$
    $ID_{seed} \leftarrow < Pk, Seed, ID_{pos}, Sig_{sk} >$
    **if** $e\%\delta_{MaxTerm} == 1$ $\&\&hash \leq ctx.w_i/ctx.W$ **then**
      $ctx.List_{ID}[Seed][e][P_{consensus}].append(ID_{seed})$
  **end for**
**end if**

In order to simplify the expression, in this paper, the private key $Sk_i$ and the public key $Pk_i$ of each consensus account $ConAC_i$, the sum of the voting power $w_i$ the node owns and represents, the total voting power $W$ of the system, and other information such as system configuration are collectively referred to as the context information of $ConAC_i$, denoted as $ctx$.

The VerifyID() (Algorithm 2) is used to verify whether the consensus identity $ID_{seed}$ is in consensus committee $Committee_{seed(P_{consensus})}$ at the phase $P_{consensus}$.

**Algorithm 2** VerifyID(): Verifying A Consensus Identity Whether in the Consensus Committee

**Input:**$ID_{seed}, P_{consensus}$
**Output:**$True or False$
$< Pk, Seed, ID_{pos}, Sig > \leftarrow$ ProcessID($ID_{seed}$)
$< hash, proof, message > \leftarrow ID_{pos}$
$< P_{consensus}, e, index > \leftarrow message$
**if** !VerifySignature($Pk, Sig$) **then return** False
**if** $ctx.E_{boot}$ *then*
  **if** $\neg VerifyVRF_{Pk}(hash, proof, Seed||P_{consensus}||e)$ **then**
    **return** True
**else if** $ctx.E_{freedom}$
  **if** $\neg VerifyVRF_{Pk}(hash, proof, Seed||P_{consensus}||e||index)$
  $\&\&(hash \leq ctx.w_i/ctx.W)$**then**
    **return**$True$
**end if**
**return** Flase

### *E. CONSENSUS*

At the same time as the formation of the consensus committee, the consensus begins accordingly. The consensus is divided into two phases $\sigma(Phase_{consensus}) \in (P_{vote}, P_{commit})$. At the $P_{vote}$, the selected members of $Committee_{seed(vote)}$ shall select a *TB* to vote, and all consensus nodes collect the vote results. At the $P_{commit}$, the members of committees $Committee_{seed(commit)}$ will start the *commit* voting based on the collected vote results, and all nodes collect the *commit* voting results. If a node counts *commit* voting results exceeds the threshold $\tau_{good}$, the consensus is reached on this node.

In a strongly synchronous network, it is optimal to reach a consensus within the time of $2 \times \delta_{term}$. The message propagation complexity is $o(C_P^2)$, where $C_P$ is the actual size of the consensus committee. It is worth noting that in the $E_{boot}$, each node has a unique identity for consensus, $C_P = C_B$; however in the $E_{freedom}$, each node has multiple consensus identities based on its voting power, and these consensus identities of the node are combined and then broadcasted. At this time, $C_P \approx C_E$.

In a weakly synchronous network environment, if the member of $Committee_{seed(vote)}$ does not receive enough votes, the $Committee_{seed(vote)}$ continues to vote, as shown in Algorithm (3, 4, 5), and a consensus can be reached according to Lemma 3 and 4. If the consensus has not yet been reached in the limited term $\delta_{MaxTerm}$, it will be suspended. After a certain period of time, the consensus will restart. When the strong synchrony comes, the reaching consensus can be guaranteed.

**Algorithm 3** PANDA_CONSENSUS()

**Input:** $ctx, Seed$
**Output:** $M_{type}, e, H_{TB}$
$e \leftarrow 1$; $H_{selectedTB} \leftarrow Empty$; $H_{TB} \leftarrow Empty$; $List_{TB} \leftarrow \{\}$
**while** $e \leq \delta_{MaxTerm}$ **do**
  $List_{TB} \leftarrow$ CollectTBlocks($H_{PRE}$)
  $H_{selectedTB} \leftarrow$ ForkSelection($ctx, List_{TB}$)
  CommitteeMsg($ctx, H_{PRE}, H_{selectedTB}, P_{vote}, e$)
  $H_{TB} \leftarrow$ CountMsg($ctx, H_{PRE}, P_{vote}$)
  **if** $H_{selectedTB} == H_{TB}$ **then**
    CommitteeMsg($ctx, H_{PRE}, H_{TB}, P_{commit}, e$)
  $H_{TB} \leftarrow$ CountMsg($ctx, H_{PRE}, P_{commit}$)
  **if** TIMEOUT $\neq H_{TB}$ **then**
    **return** $< COMMIT, e, H_{TB} >$
  $e + +$
**end while**

The consensus node monitors the presence of the *Spy* identity (definition in section B of part VII ) during the counting process, and *Evd* will be collected and broadcasted to the entire networks as soon as the *Spy* identity is discovered (e.g. an identity $ID_i$ discovers that an identity $ID_j$ has voted for both $H_a$ and $H_b$ in a certain term of the consensus voting, then the evidence $Evd < Pk_i, Pk_j, \{H_a, Sig_a\}, \{H_b, Sig_b\}, Sig >$ will be saved and broadcasted). Upon the knowledge and verification of other nodes, the node corresponding to the *Spy* identity is blacklisted, and the voting of the blacklisted node is ignored in the following consensus. The consensus deposit of the node is deducted at the end of the consensus. Therefore, the best choice for a malicious node is to select only one *TB* to vote, and try to delay the consensus time as much as possible, or not to vote at all.

The CommitteeMsg() (Algorithm 4) is the algorithm used by members of the $Committee_{seed}$ to send messages. The type of the sent message $\sigma(M_{type})$ is divided into $(M_{vote}, M_{commit})$ according to the phase in which the consensus is located, where $M_{vote}$ is the message used by the phase $P_{vote}$ to vote for the selected *TB*, while $M_{commit}$ is the message used by the phase $P_{commit}$ to commit the *TB* based on the collected vote results.

**Algorithm 4** CommitteeMsg(): Broadcasting Messages by Committee Members

**Input:** $ctx, Seed, H_{TB}, P_{consensus}, e$
$M_{type} = GetMsgType(P_{consensus})$
$index = e/\delta_{MaxTerm} + 1$
**if** $List_{ID_{seed}} \leftarrow ctx.List_{ID}[Seed][P_{consensus}][index]$
!isEmpty($List_{ID_{seed}}$) **then**
  SendMsg($M_{type}, H_{TB}, List_{ID_{seed}}, Sig_{ctx.sk}$)
**end if**

The CountMsg() (Algorithm 5) is used by the consensus nodes to collect and count the number of messages. If the received message amount exceeds the threshold $\tau_{good}$, the hash of the corresponding transaction block and its term are returned; if the threshold is not exceeded within $\delta_{term}$, *TIMEOUT* is returned.

**Algorithm 5** CountMsg(): Counting Messages

**Input:** $ctx, Seed, M_{type}$
**Output:** $H_{TB}$ or *TIMEOUT*
$start \leftarrow$ Time(); $counts \leftarrow \{\}$; $voters \leftarrow \{\}$
$msgs \leftarrow CollectGobalMsgs(M_{type}).iterator()$
**while** *True* **do**
  **if** Time() $> start + ctx.\delta_{Term}$ **then** TIMEOUT
  $m \leftarrow msgs.next()$
  $P_{consensus} \leftarrow$ GetPhase($M_{type}$)
  $< ID_{seed}, H_{TB} > \leftarrow$ ProcessMsg($m$)
  **if** !VerifyID($ID_{seed}, P_{consensus}$) **then continue**
  **if** $ID_{seed} \in voters[ID_{seed}.e][M_{type}]$ **then continue**
  $counts[ID_{seed}.e][M_{type}][H_{TB}] + +$
  **if** $counts[ID_{seed}.e][M_{type}][H_{TB}] \geq \tau_{good}$ **then**
    **return** $H_{TB}$
**end while**

## VII. ATTACK VECTORS AND SECURITY ANALYSIS

### *A. ATTACK VECTORS*

#### 1) DOUBLE SPENDING ATTACK

Double-spending is the core problem faced by any cryptocurrency, where an adversary holding \$1 gives his \$1 to two

**TABLE 2.** **Potential problems in each step of PANDA protocol and the corresponding lemmas which resolve them.**

| Step | Potential problems | Lemma |
|---|---|---|
| Identity Setup | Too many malicious identities | Lem.1 |
| Committee Formation | Bias the committee distribution | Lem.2 |
| Consensus | Cannot reach consensus | Lem.3,4 |

different users [15]. DLattice prevents double-spending by Fork Penalty and PANDA consensus.

First, each transaction block $TB$ is required to reserve the Fork Penalty (FP) at their creation. When the fork occurs, the honest node selects a $TB$ from the fork set, and then resolves the fork on the basis of PANDA consensus. All participating identities in the consensus have the opportunity to obtain the FP of the $TB$. The algorithm for allocation of fork penalty is shown as Algorithm 6. It is worth mentioning that the FP will be obtained only if the XOR distance between the consensus identity $ID_{seed}$ and the previous hash $H_{PRE}$ is less than the threshold $\tau_{penalty}$, so the more $DLT$ the consensus account holds, the greater probability to obtain the penalty.

**Algorithm 6** ForkPenaltyAllocationCheck(): Allocation of Fork Penalty

**Input:** $ctx$, $Seed$, $e$
**Output:** True or False
$flag \leftarrow$ False
$List_{ID_{seed}} \leftarrow ctx.List_{ID}[Seed][e]$
**for** $i = 0$; $i < Len(List_{ID_{seed}})$; $i++$
  $dist = List_{ID_{seed}}[i] \oplus Seed$
  **if** $dist \leq ctx.\tau_{penalty}$ **then** $flag \leftarrow$ True
**end for**
**reture** $flag$

### 2) SYBIL ATTACK

If there is no trusted public key infrastructure in a system, a malicious node can simulate many virtual nodes, thereby creating a large set of sybils. An entity could create hundreds of nodes on a single machine [32].

However, since the identities of these nodes in consensus process are created in proportion to their account balances, adding extra nodes into the network will not gain an attacker extra vote. Therefore, sybil attack will bring no advantage.

### 3) DDOS ATTACK

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic [33].

According to the previous analysis, in a strongly synchronous network, the consensus is reached within the first term. At this time, the member of $Committee_{seed(vote)}$ and $Committee_{seed(commit)}$at the consensus phase are non-interactively selected based on VRF, which has a posteriority to prevent DDoS attacks and collusion among committee members. If the consensus is not reached in first term (possibly due to the randomness of generation of the committee or a weakly synchronous network environment), the $Committee_{seed(vote)}$ may indeed suffer DDoS attacks due to exposure. However, first, the attack will not affect other consensus committee to resolve other forks. Second, the development of the entire system will not be affected because only the consensus committee resolving the fork generated by malicious users will suffer from DDoS attack. Finally, thanks to the existence of the FP, committee members who have suffered DDoS attacks will eliminate the DDoS attack as soon as possible to reach consensus, so as to obtain rewards.

### 4) FLUCTUATION OF NODES

Generally, the amount of consensus nodes will show a trend of growth over time, as shown by the green line in Fig. 6. Before the time $t_1$, the system is in the $E_{boot}$, each $Candidate_{seed}$ only has a consensus identity; when the time is at $t_2$, the system enters the $E_{freedom}$ from the $E_{boot}$, each $Candidate_{seed}$ may have multiple consensus identities; but when the time reaches $t_3$ (on the blue curve), and the nodes in the system have less than $C_B$ for various reasons, the system is still in the $E_{freedom}$ (and it will not go back to the $E_{boot}$). If the remaining active honest nodes still have the voting power of $h$, although the actual amount of nodes is less than the size $C_B$ of the $BootCommittee$, the actual amount of generated identities $C_P$ still satisfy $C_E \approx C_P$. According to Lemma 2, at most $C_P/3-1$ consensus identities are controlled by the Byzantine node, so that an effective consensus can still be reached.

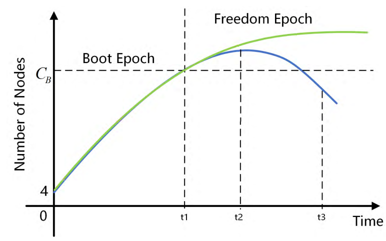



**FIGURE 6.** **Schematic diagram of fluctuation of Nodes.**

## B. SECURITY ANALYSIS

In this section, we provide security analysis for how DLattice prevents potential threats and works securely based on several assumptions clarified in Section IV. We also discuss how the byzantine adversary gains no significant advantage.

***Definition Spy.*** *If the behavior of an identity is dishonest and is discovered, we call this identity a Spy, and we can obtain evidence based on dishonest behavior, which we callEvd, like voting for*$H_a$*at the same time as voting for*$H_b$*at the phase*$P_{vote}$*, or other behaviors like that.*

***Definition Ballot.****If a node receives at least*$\tau_{good}$*votes at the phase*$P_{vote}$*, we call it the observation of a*$Ballot$*. And the node can only vote or commit this*$Ballot$*at the phase*$P_{vote}$*or*$P_{commit}$ *in the later term [34].*

*Lemma 1 (Consensus Identity Generation):* In the Consensus Identity Setup phase of the $E_{freedom}$, each consensus identity is generated based on the voting power held by the node. The generation of consensus identity and the number of calculating attempts obey the exponential distribution.

*Proof:* Assume that $\beta$ indicates the number of attempts required to generate a consensus identity; $\theta$ indicates the probability of generating a consensus identity at one attempt. The probability of generating a legal consensus identity within $\beta$ attempts is:

$$P\{x \leq \beta\} = 1 - P\{x > \beta\} = 1 - (1-\theta)^{\beta}$$

$= 1 - e^{\beta \log(1-\theta)}$.

Considering $\theta \ll 1$ (in general), thus $\log(1-\theta) \approx -\theta$, and we can achieve:

$$P\{x \leq \beta\} \approx 1 - e^{-\theta\beta}.$$

Therefore, the consensus identity generation and the required number of attempts obey the exponential distribution. If we set $\theta = w_i/W$, where $w_i$ refers to the voting power of the node, $W$ refers to the total voting power, then:

$$P\{x \leq \beta\} = 1 - e^{-\frac{w_i\beta}{W}}.$$

When the total voting power $W = 12000$ and the voting power of two consensus nodes are $w_i = 10$ and $w_j = 20$ respectively, the probability of generating a legal consensus identity in the $\beta = 400$ calculations is shown in Fig. 7. It's shown in the figure that when $\beta$ is a fixed value, the greater the voting power, the greater the probability of generating a legal identity.

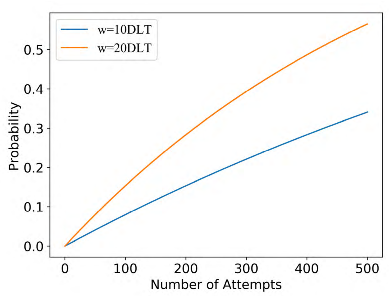



**FIGURE 7. Generation of consensus identities and the number of attempts obey an exponential probability distribution, where the orange curve represents $w_j = 20DLT$, and blue curve illustrates $w_i = 10DLT$.**

*Lemma 2 (Number of Consensus Identities):* In the Committee Formation phase of the $E_{freedom}$, the candidate consensus nodes generate consensus identities based on their voting power and establish a consensus committee $Committee_{seed(P_{consensus})}$. The system guarantees that no multiple consensus will be reached in the consensus committee.

*Proof Sketch:* According to Lemma 1, the candidate consensus nodes generate consensus identities base on their voting power. The probability of a consensus account owned voting power $w_i$ generating $k$ consensus identities within $\beta$ calculations is:

$$P\{x = k\} = \binom{\beta}{k}(\frac{w_i}{W})^k(1-\frac{w_i}{W})^{\beta-k}.$$

The expectation is $E = \beta w_i/W$. According to Assumption 3, the $DLT_{good}$ held by honest node and $DLT_{total}$ of the systems always satisfy:

$$h = DLT_{good}/DLT_{total}.$$

If the total voting power is $W = DLT_{total}$, the voting power of the honest nodes is $w_{honest} = DLT_{good}$, the voting power of the malicious nodes is $w_{adversary} = DLT_{bad}$. The consensus identity expectation generated by the honest nodes is $E_{bad} = \beta w_{adversary}/W$, the consensus identity expectation generated by the malicious nodes is $E_{bad} = \beta w_{adversary}/W$, so the consensus identity that the system expects to generate is $C_E \approx \beta$, and the honest identity accounts for about $h$ of the total.

Assume that the maximum and minimum identities generated by consensus nodes, the honest nodes and the malicious nodes are $all_{\max}$, $all_{\min}$, $h_{\max}$, $h_{\min}$, $a_{\max}$ and $a_{\min}$ respectively. We can list equations as follows:

$$\begin{cases} P_{unit} = \frac{1}{W}, P_{honest} = w_{honest} \times P_{unit}, P_{adversary} \\ \quad = w_{adversary} \times P_{unit} \\ P\{x < h_{\max}\} = \sum_{k=0}^{h_{\max}} \binom{\beta/P_{honest}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{honest}-k} \\ P\{x < a_{\max}\} = \sum_{k=0}^{a_{\max}} \binom{\beta/P_{adversary}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{adversary}-k} \\ P\{x < all_{\max}\} = \sum_{k=0}^{all_{\max}} \binom{\beta/P_{unit}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{unit}-k} \\ P\{h_{\min} < x \leq \beta/P_{honest}\} = \sum_{k=h_{\min}}^{\beta/P_{honest}} \binom{\beta/P_{honest}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{honest}-k} \\ P\{a_{\min} < x \leq \beta/P_{adversary}\} = \sum_{k=a_{\min}}^{\beta/P_{adversary}} \binom{\beta/P_{adversary}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{adversary}-k} \\ P\{all_{\min} < x \leq \beta/P_{unit}\} = \sum_{k=a_{\min}}^{\beta/P_{unit}} \binom{\beta/P_{unit}}{k} \\ \quad (P_{unit})^k(1-P_{unit})^{\beta/P_{unit}-k} \end{cases}$$

And then we set:

$$\begin{cases} P\{x < h_{\max}\} \geq 1 - 2^{-\lambda} \\ P\{x < h_{\min}\} \geq 1 - 2^{-\lambda} \\ P\{h_{\min} < x \leq \beta\} \geq 1 - 2^{-\lambda} \\ P\{a_{\min} < x \leq \beta\} \geq 1 - 2^{-\lambda} \\ P\{x < all_{\max}\} \geq 1 - 2^{-\lambda} \\ P\{all_{\min} < x \leq \beta/P_{unit}\} \geq 1 - 2^{-\lambda} \end{cases}$$

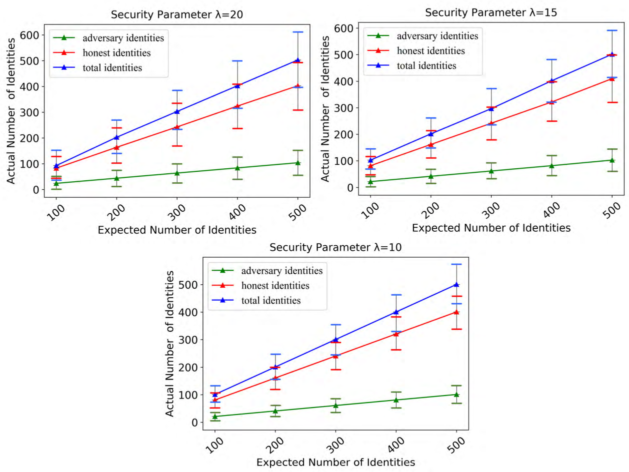



**FIGURE 8. The maximum and minimum number of all identities *all*, honest identities *h*, and malicious identities *a* that the system may generate with different security parameters.**

Figure 8. The maximum and minimum number of all identities *all*, honest identities *h*, and malicious identities *a* that the system may generate with different security parameters.

When $\beta$ is equal to 100, 200, 300, 400 and 500, respectively, the calculation results of different security parameters $\lambda$ are shown in Fig. 8. At the same time, the relationship between $all_{\max}$, $h_{\min}$ and $a_{\max}$ should be satisfied as follows:

$$\begin{cases} ID_{good} > 2 \times a_{\max} \\ ID_{good} \leq h_{\min} \\ 2 \times ID_{good} > all_{\max} \end{cases}$$

The system requires at least $ID_{good}$ honest identities to avoid multiple consensus results being reached during the consensus process. The value of $ID_{good}$ depends on the security parameters. When the security parameter is $\lambda = 20$, the most ideal result is $\beta = 500$ and $ID_{good} = 306$; when $\lambda = 15$, the result is $\beta = 400$ and $ID_{good} = 250$; when$\lambda = 10$, $\beta = 200$ and $ID_{good} = 123$. The values above ensure that no multiple consensus results will be reached at the phase $P_{commit}$.

*Lemma 3 (Proof of Safety):* In the consensus process, if the consensus identities reach a consensus on a $TB_{send}$ in the fork set $\{TB_{send}, .., TB^{'}_{send}\}$, no consensus will be reached on the other $TB^{'}_{send}$.

*Proof:* $C_P$ represents the actual size of the consensus committee; $X$ indicates the number of honest identities that have committed $TB_{send}$ in the $e$ term, while $1 \leq X \leq ID_{good}$; at the phase $P_{vote}$ in the $e+1$ term, $X$ honest identities have to continue to vote as same as in the $e$ term due to the existence of $Ballot$; $Y$ stands for the number of malicious identity which can do anything, and $Y < ID_{good}/2$; $Z$ indicates the number of remaining identities. And these parameters satisfy $X+Y+Z = C_P$; $Z_0$ indicates the number of identities in the remaining identities that have voted at the phase $P_{vote}$ in term e; $Z - Z_0$ indicates the number of identities that have not yet voted;

Once $X$ identities have committed $TB_{send}$, the remaining $Y + Z$ identities will not reach a consensus on the $TB^{'}_{send}$, that is, to prove:

$$Y + Z - Z_0 < ID_{good}.$$

We can prove it by contradiction, assume that:

$$Y + Z - Z_0 \geq ID_{good} \Rightarrow Y + Z - Z_0 + X \geq ID_{good} + X$$

$$\Rightarrow C_P - Z_0 \geq ID_{good} + X \Rightarrow X + Z_0 \leq C_P - ID_{good}.$$

Since $C_P = Y + ID_{good} < 3 \times ID_{good}/2$, then $X + Z_0 < ID_{good}/2$.

Assume that all malicious identities have voted on the $TB_{send}$ in the term e, $X + Y + Z_0 > ID_{good}$, also because $Y < ID_{good}/2$, then $X+Z_0 > ID_{good}/2$, which is inconsistent with the assumption; if some malicious identities $Y^{'}$ have voted on the $TB_{send}$ in the term e, $X + Y^{'} + Z_0 \geq ID_{good}$, also because $Y^{'} < Y < ID_{good}/2$, so $X + Z_0 > ID_{good}/2$, which is also inconsistent with the assumption.

*Lemma 4 (Proof of Liveness):*In the consensus phase, if a consensus identity is locked on *Ballot* of a *TB* in the *e* term, when term $e^{'} > e$, if the node find a new $Ballot^{'}$, and the *Ballot* in the *e* term is unlocked while the new $Ballot^{'}$ is locked, thus ensuring the continuation of the consensus.

*Proof Sketch:* In the $P_{commit}$, due to the existence of *Ballot*, some nodes may commit for *Ballot* while the other nodes is committing for $Ballot^{'}$, and the votes from both parties are just equal that led to the failure of reaching a consensus. At the $P_{vote}$, if the node finds the $Ballot^{'}$ of higher term, it indicates that the current system is more inclined to reach a consensus for $Ballot^{'}$ of higher term, so the node unlocks the *Ballot* of lower term and locks $Ballot^{'}$ of the higher term. And at the $P_{commit}$, the nodes will commit the new locked $Ballot^{'}$.

## VIII. IMPLEMENTATION AND EVALUATION

We implement DLattice and the goals of our evaluation are twofold. We first measure the latency and throughput of DLattice when the network size increases. The second goal is to compare DLattice to other related consensus protocols including Bitcoin, Ethereum and PBFT, etc.

### A. IMPLEMENTATION

We implement a prototype of DLattice in Golang [35], consisting of approximately 4000 lines of code. We implement a gossip network by using go-libp2p library (go-libp2p-pubsub) [36] where each user selects a small random set of peers to gossip messages to. Elliptic Curve Cryptography (ECC) encryption algorithm is used for asymmetric encryption while the symmetrical algorithm adopts AES algorithm. SHA-256 is a cryptographic hash function for us to calculate hash value. And we use the VRF outlined in Goldberg [37]. The signature algorithm adopts Elliptic Curve Digital Signature Algorithm (ECDSA).

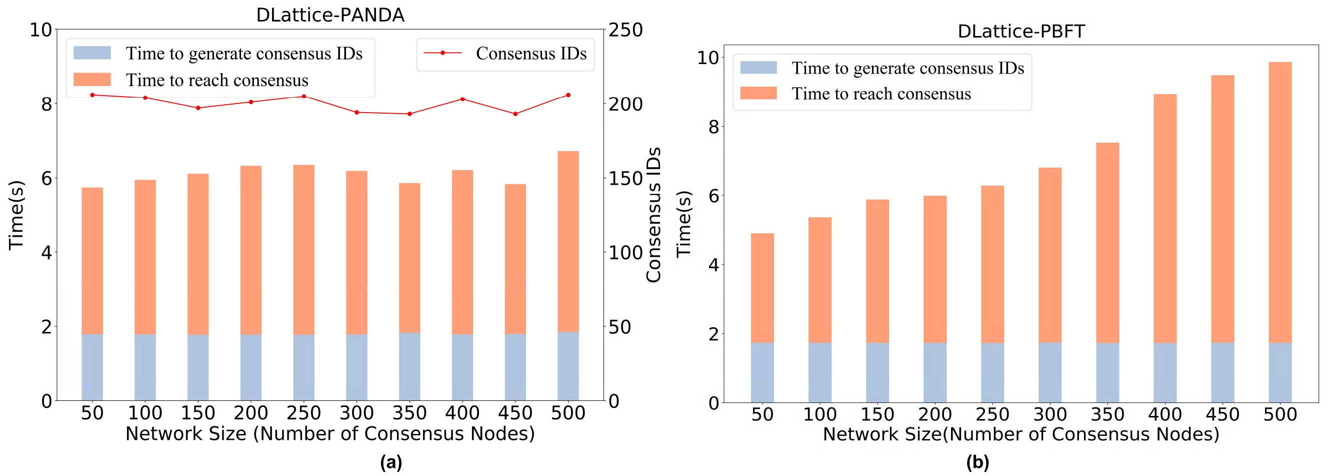



**FIGURE 9.** Latency to reach consensus using PANDA (a) and PBFT (b) respectively with 50 to 500 consensus nodes. The broken line in (a) represents the number of identities participating in the PANDA consensus.

**TABLE 3.** Implementation Parameters.

| Parameter | Meaning | Value |
|---|---|---|
| $\lambda$ | Security Parameter | 10 |
| $h$ | Weight of the $DLT$ held by the honest node in the total $DLT$ | 4/5 |
| $DLT_{total}$ | Total $DLT$ in DLattice | 12000 |
| $DLT_{init}$ | Initially allocated amount of $DLT$ per boot node | 60 |
| $C_B$ | Size of a boot committee | 200 |
| $C_E$ | Expected size of a consensus committee | 200 |
| $\tau_{good}^{boot}$ | Threshold of the honest identities in total identity in the Boot Epoch | 2/3 |
| $\tau_{good}^{freedom}$ | Number of the honest identities in the Freedom Epoch | 123 |
| $\delta_{term}$ | Time of each term | 20s |

Table 3 shows the parameters in implementation of prototype of DLattice; we mathematically validate these parameters $\lambda$, $C_E\tau_{good}^{freedom}$, etc. in Section VII. $h = 4/5$ means that an adversary would need to control 20% of DLattice's currency in order to create a fork. $\delta_{term}$ should be high enough to allow users to receive messages from committee members, but low enough to allow DLattice to make progress if it does not hear from sufficiently many committee members. We conservatively set $\delta_{term}$ to 20 seconds.

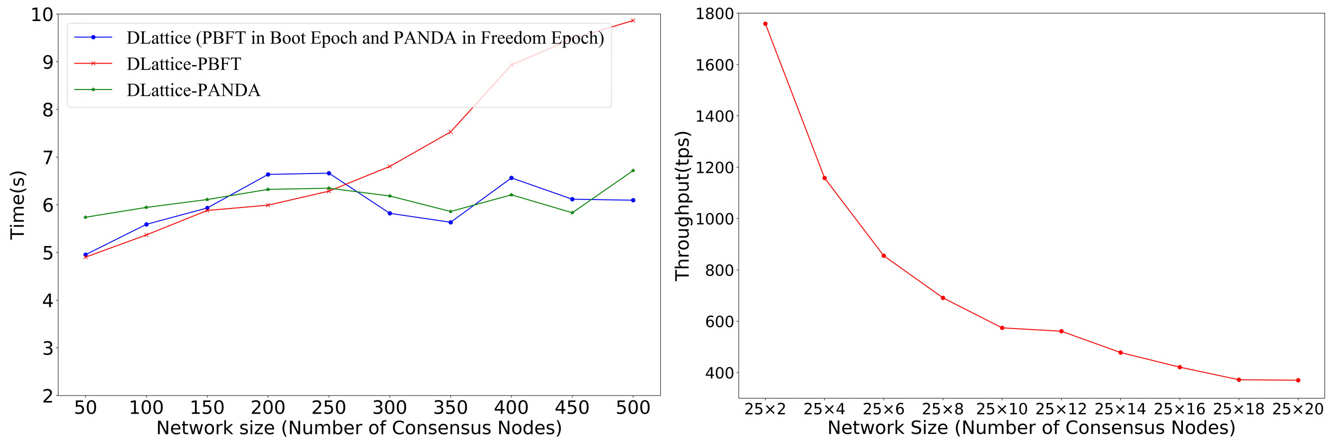
### B. EVALUATION

We run several experiments with different settings on AliCloud ECS servers to measure the latency and throughput of DLattice. We vary the number of consensus nodes in the network from 50 to 500, using up to 25 ECS instances. Each AliCloud ECS instance is shared at most by twenty nodes, and has 4 AliCloud vCPUs and 8 GB of memory.

#### 1) LATENCY

Latency is the amount of time that it takes from the creation of a transaction until the initial confirmation of it being accepted by the network [38]. The latency of Bitcoin and Ethereum is 576.4 seconds (from Block Height 556800 to 556810) [39] and 12 seconds (from Block Height 7002602 to 7002612) [40] respectively in the livenet.

The latency of transaction in DLattice is instantaneous, so we just consider the consensus latency in this section. We implement a consensus algorithm similar to PBFT for Boot Epoch of DLattice. The consensus latency of the algorithm is shown in Figure 9(b), where the consensus latency includes the time to generate consensus identity and time to reach consensus. As the number of consensus nodes increases from 50 to 500, the time continues to rise. Similarly, PANDA is implemented for Freedom Epoch of DLattice. During the experiment, when the number of nodes increases from 50 to 500 and the corresponding voting weight decreases from 240 to 24, the consensus latency is shown in Figure 9(a). As shown in Figure 10(a), since the size of the consensus message of PANDA (about 0.86 kb) is larger than that of PBFT (about 0.4 kb) messages, the latency of PBFT is smaller than that of PANDA when the consensus nodes are less than 250. As the number of nodes increases, the number of identities participating in the consensus in PANDA is oscillating around the expected consensus identities, as shown in the broken line in Figure 9(b), and the consensus identity in PBFT increases with the number of nodes, so the difference between two latency is growing larger. Figure 10(a) shows the comparison among the latency of DLattice with Boot Epoch and Freedom Epoch (PBFT is used when consensus nodes are less than 200 in the Boot Epoch, and PANDA is used when nodes are more than 200 in the Freedom Epoch), the latency of DLattice-PBFT (PBFT only) and that of DLattice-PANDA (PANDA only).




(a) (b)

**FIGURE 10.** **(a) Latency to reach consensus in DLattice with 50 to 500 consensus nodes; The blue curve represents DLattice, the red curve represents DLattice-PBFT, and the green curve represents DLattice-PANDA. (b)Throughput of DLattice with 50 to 500 consensus nodes.**

**TABLE 4.** **Comparison between DLattice and existing blockchain protocols in academia and industry.**

| | Nakamoto | PoS | | DPoS | BFT | DAG | DLattice |
|---|---|---|---|---|---|---|---|
| Candidate | Bitcoin [12]<br>Ethereum [13]<br>Bitcoin-NG [14] | Algorand [15] | Ouroboros [40]<br>Dfinity [46] | EOS [47]<br>BitsShares [48] | IBM Blockchain [49]<br>Tendermint [24] | Nano [26] | This work |
| Identity-less | Yes √ | No | | No | No | No | Yes √ |
| Bandwidth | Constant√ | Constant√ | | Constant√ | $O(n^2)$ | Constant√ | Constant√ |
| Posteriority | No | Yes √ | No | No | No | No | Yes √ |
| Storage-pruned | No | No | | No | No | Yes √ | Yes √ |

### 2) THROUGHPUT

One of the end goals of blockchain is to replace the current infrastructure (like financial back-end of many institutions around the world, which handles thousands of transactions per second (TPS)), it will need to scale to meet and/or exceed the TPS to prove its viability. A higher throughput will also open the doors to more interesting and intensive applications of blockchain technology [41]. The Bitcoin network processes up to 3 TPS (from Block Height 556800 to 556810) [39] and Ethereum processes up to 127 TPS (from Block Height 7002602 to 7002612) [40] in the livenet. And Nano claims that it has theoretical 7000 TPS [42], and experimental 756 TPS in the beta network stress test [43].

From the foregoing, transaction block types of DLattice include $TB_{send}$, $TB_{receive}$, $TB_{auth}$ etc., where $TB_{receive}$ has an average size of about 0.5 kb, and the size of the $TB_{auth}$ varies with the number of authorizations. and average size of $TB_{send}$ is about 0.7 kb. Therefore, during the experiment, the time required to receive 25,000 sending transaction blocks is counted. All numbers are averaged after 10 times. We start with a network of 50 consensus nodes, and then rise to 500 nodes in the last setting. The throughput of DLattice is shown in Figure 10(b). As the number of DLattice nodes deployed per ECS instance increases, the throughput of DLattice is decreasing due to hardware constraints. However, since the sending of transaction block of each account are asynchronous with other accounts so it is unnecessary to wait for miners to pack transactions like traditional blockchains. Although it does not reach 7000 TPS in Nano (Because Nano's block is smaller, and DLattice records more information about data tokenization), it is still close to 1200 TPS when less than four nodes are deployed per ECS instance (However, each computer is likely to deploy only one DLattice node in a real environment). We will further optimize its throughput in the future experiments and practical scenarios.

### 3) COMPARISON TO RELATED SYSTEMS

The comparison between DLattice and the existing blockchain consensus protocols is shown in Table 4. Compared with the traditional Nakamoto consensus algorithm, the DLattice with PANDA consensus solves the problem of high energy consumption; compared with the traditional BFT consensus, the consensus identity has a posteriority, that is, the randomly elected consensus committee is able to prove its identity without revealing it in advance. In addition, as the number of nodes increases, there is no significant change in network bandwidth consumption. The round-based Algorand lacks economic incentives, and the signature data is large, which has strict requirements on network bandwidth. The chain-based Ouroboros [44] is established in a strong synchrony network and the slot leader has been selected in advance (these drawbacks have been improved by [45]). Inspired by Nano, DLattice builds a Double-DAG architecture that is dedicated to the tokenization of data. Compared with Nano, DLattice slightly sacrifices the processing speed and throughput rate of transactions, but the random selection of consensus identity reduces the risk of DDoS attacks and the possibility of collusion among nodes. Moreover, if the consensus process is in a strong synchrony network, the consensus can be reached within the first term.

## IX. CONCLUSION AND FUTURE WORK

In this paper, we propose a new permission-less blockchain, called DLattice, with a Double-DAG architecture where each account has its own Account-DAG and all accounts make up a greater Node-DAG structure. DLattice parallelizes the growth of each account's Account-DAG, each of which is not influenced by other accounts' irrelevant transactions, resulting in fast transactions with minimal overhead. The core of DLattice is DPoS-BA-DAG (PANDA) protocol which helps users reach consensus with low latency only when the forks are observed. Based on proposed DLattice structure, we introduce a series of methods for data tokenization, including data assembling, data anchoring and data authorization. DLattice tokenizes data as on-chain assets for sale and transaction, making them circulate and transfer securely and efficiently. Through security analysis, we demonstrate DLattice can prevent some attack vectors such as Double-Spend, Sybil attack, etc. Experimental results show that DLattice reaches a consensus in 10 seconds, achieves desired throughput, and incurs almost no penalty for scaling to more users.

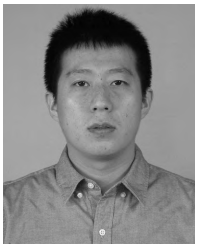
The shortcomings of this paper are that i) the current DLattice prototype only anchors the digital fingerprint of the data asset, while the raw data is stored in the IPFS, but due to the lack of incentives, the data may be lost with the accidental offline of the IPFS nodes. The function of the consensus node shall be optimized in the following study, making it not only participate in the consensus but also has the ability to store data assets; ii) smart contracts are not currently supported; iii) in the process of consensus, if the consensus is not reached in the first term, and the identity of the corresponding consensus committee membership will be exposed and vulnerable to DDoS attacks. These issues are the focus of research in the following work.

In the future studies, we consider introducing DLattice to healthcare and the Internet of Things to achieve the tokenization of medical data and IoT data. One possible application is to manage chronic diseases by tokenizing the medical examination data, health data collected by wearable devices and exercise prescription data issued by doctors based on our previous experience in the field of health informatics. In this way, the physiological and exercise data of users are effectively protected while being asserted, and the data can be efficiently shared and transferred among scientific research institutions, hospitals, health equipment manufacturers, and even insurance companies.

## REFERENCES

[1] E. Zhou. *China's Biggest Hotel Operator Leaks 500m Customer Records in Data Breach*. Accessed: Aug. 12, 2018. [Online]. Available: https://www.mingtiandi.com/real-estate/finance-real- estate/huazhu-hotels-leaks-500m-customer-records-in-data-breach/
[2] D. Ingram. *Facebook Says Data Leak Hits 87 Million Users, Widening Privacy Scandal*. Accessed: Aug. 15, 2018. [Online]. Available: https://www.reuters.com/article/us-facebook-privacy/facebook-says-data-leak-hits-87-million-users-widening-privacy-scandal-idUSKCN1HB2CM
[3] *Braggadocio, Information Control, and Fear: Life Inside a Brigham Stem Cell Lab Under Investigation*. Accessed: Oct. 20, 2018. [Online]. Available: https://retractionwatch.com/2014/05/30/braggadacio-information-control-and-fear-life-inside-a-brigham-stem-cell-lab-under-investigation/
[4] T. Zhou, X. Li, and H. Zhao, "EverSSDI: Blockchain-Based Framework for Verification, Authorization and Recovery of Self-Sovereign Identity using Smart Contracts," *Int. J. Comput. Appl. Technol.*, to be published.
[5] N. Sheng *et al.*, "Data capitalization method based on blockchain smart contract for Internet of Things," *J. Zhejiang Univ. (Engineering Science)*, vol. 52, no. 11, pp. 2150–2153, Nov. 2018.
[6] G. Wood. *Ethereum: A Secure Decentralized Generalized Transaction Ledger*. Accessed: Sep. 20, 2018. [Online]. Available: https://ethereum.github.io/yellowpaper /paper.pdf
[7] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *J. Amer. Med. Inform. Assoc.*, vol. 24, no. 6, pp. 1211–1220, 2017.
[8] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," in *Proc. AMIA Annu. Symp.*, 2017, pp. 650–659.
[9] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018. doi: 10.1109/ACCESS.2018.2851611.
[10] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[11] *Multichain: Open Platform for Blockchain Applications*. Accessed: Sep. 11, 2018. [Online]. Available: https://www.multichain.com/
[12] P. Serguei. *The Tangle*. Accessed: Sep. 2018. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
[13] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Sep. 20, 2018. [Online]. Available: https://bitco.in/pdf /bitcoin.pdf
[14] I. Eyal, A. E. Gencer, E. G. Sirer and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Conf. Netw. Syst. Design Implement.* Berkeley, CA, USA: USENIX Association, 2016, pp. 45–59.
[15] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Oper Syst. Princ.*, 2017, pp. 51–68.
[16] S. King and S. Nadal. (2012). *PPcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. Accessed: Sep. 20, 2018. [Online]. Available: https://peercoin.net /assets/paper/peercoin-paper.pdf
[17] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," In *Proc. 3rd Symp. Oper. Syst. Design Implement.*, Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.
[18] 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.*, 2016, pp. 17–30.
[19] B. Group. *Proof of Stake versus Proof of Work White Paper*. Accessed: Sep. 25, 2018. [Online]. Available: https://bitfury.com/content/downloads/pos-vs-pow-1.0.2.pdf
[20] *Bitcoin Average Confirmation Time*. Accessed: Mar. 25, 2018. [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[21] V. Zamfir. *Introducing Casper 'the Friendly Ghost'*," Accessed: Nov. 2018. [Online]. Available: https://ethereum.github.io/blog/ 2015/08/01/introducing-casper-friendly-ghost/
[22] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.
[23] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[24] E. Buchman, J. Kwon, and Z. Milosevic. (2018). "The latest gossip on BFT consensus." [Online]. Available: https://arxiv.org/abs/1807.04938

[25] L. Baird. *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*. Accessed: Sep. 05, 2018. [Online]. Available: http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf
[26] C. LeMahieu. *Nano: A Feeless Distributed Cryptocurrency Network*. Accessed: Aug. 6, 2018. [Online]. Available: https://nano.org/en/whitepaper
[27] X.-P. Min, Q.-Z. Li, L.-J. Kong, S.-D. Zhang, Y.-Q. Zheng, and Z.-S. Xiao, "Permissioned blockchain dynamic consensus mechanism based multi-centers," *Chin. J. Comput.*, vol. 41, no. 5, pp. 1005–1020, 2018. doi: 10.11897/SP.J.1016.2018.01005.
[28] C. Wong. *Patricia Tree*. Accessed: Mar. 25, 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree
[29] *Red-Black Merkle Tree*. Accessed: Nov. 17, 2018. [Online]. Available: https://github.com/amiller/redblackmerkle.
[30] J. Benet. *IPFS-Content Addressed, Versioned, P2P File System*. Accessed: Oct. 14, 2018. [Online]. Available: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf
[31] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, New York, NY, USA, Oct. 1999, pp. 120–130.
[32] J. R. Douceur, "The Sybil attack," in *Proc. 1st Int. Workshop Peer-Peer Syst. (IPTPS)*, Cambridge, MA, USA, Mar. 2002, pp. 251—260.
[33] *DDOS*. Accessed: Oct. 20, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Denial-of-service_attack
[34] D. Ojha. *Byzantine Consensus Algorithm*. Accessed: Sep. 11, 2018. [Online]. Available: https://github.com/tendermint/tendermint/wiki/Byzantine-Consensus-Algorithm
[35] *Golang*. Accessed: Jan. 3, 2019. [Online]. Available: https://golang.org/
[36] *Libp2p*. Accessed: Dec. 15, 2018. [Online]. Available: https://github.com/libp2p
[37] *A VRF implementation in golang*. Accessed: Dec. 22, 2018. [Online]. Available: https://github.com/r2ishiguro/vrf/
[38] A. Grigorean. *Latency and Finality in Different Cryptocurrencies*. Accessed: Jan. 4, 2019. [Online]. Available: https://hackernoon.com/latency-and-finality-in-different-cryptocurrencies-a7182a06d07a
[39] *Bitcoin Explorer*. Accessed: Jan. 3, 2019. [Online]. Available: https://btc.com/
[40] *Ethereum Explorer*. Accessed: Jan. 3, 2018. [Online]. Available: https://etherscan.io/
[41] *Zilliqa: A High Throughput Scalable Blockchain?* Accessed: Jan. 4, 2019. [Online]. Available: https://medium.com/ @curiousinvestor/zilliqa-a-high-throughput-scalable-blockchain-60e355d873c5
[42] A. Anand. *Nano Embraces Speed, Sees Transaction Rate Jump to 750 TPS*. Accessed: Jan. 4, 2019. [Online]. Available: https://ambcrypto.com/nano-embraces-speed-sees-transaction-rate-jump-to-750-tps/
[43] *Nano*. Accessed: Jan. 4, 2019. [Online]. Available: https:// nano.org/
[44] A. Kiayias, R. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2017, pp. 357–388.
[45] B. Davi *et al. Ouroboros PRAOS: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain*. Accessed: Mar. 25, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-78375-8_3
[46] T. Hanke, M. Movahedi and D. Williams. *Dfinity Whitepaper*. Accessed: Nov. 13, 2018. [Online]. Available: https://dfinity.org/static/ dfinity-consensus-0325c35128c72b42df7dd30c22c41208.pdf
[47] I. Grigg. *EOS Whitepaper*. Accessed: Oct. 15, 2018. [Online]. Available: https://eos.io/documents/EOS_An_Introduction.pdf
[48] *The Bitshares Blockchain*. Accessed: Oct. 25, 2018. [Online]. Available: https://www.bitshares.foundation/download/articles/BitSharesBlockchain.pdf
[49] I. Research. *Blockchain/DLT: A Game-Changer in Managing MNCs Intercompany Transactions*. Accessed: Oct. 28, 2018. [Online]. Available: https://www.ibm.com/think/fintech/wp-content/uploads/2018/03/IBM_Research_MNC_ICA_Whitepaper.pdf



**TONG ZHOU** received the B.S. degree in software engineering from Hubei University, Wuhan, China, in 2013, and the M.S. degree in information systems and signal processing from Anhui University, Anhui, China, in 2016. He is currently pursuing the Ph.D. degree in computer applied technology with the University of Science and Technology of China, Hefei, China. His research interests include blockchain technology, consensus algorithm, and health informatics.

**XIAOFENG LI** received the B.S. degree from Tianjin University, in 1987. He is currently a Research Professor with the Hefei Institutes of Physical Science, Chinese Academy of Sciences (CASHIPS), and a Doctoral Supervisor with the University of Science and Technology of China. He is also the Director of the Internet Network Information Center, CASHIPS, the Vice Chairman of the Hefei Branch of Association for Computing Machinery (ACM), and the Vice Chairman of the Anhui Radio Technology Association. His current research interests include blockchain technology, computer applied technology and measurement, control technology, and automation instrument.

**HE ZHAO** received the B.S. and M.S. degrees from the Nanjing University of Posts and Telecommunications, in 2007 and 2010, respectively, and the Ph.D. degree from the University of Science and Technology of China, in 2016. He has been with Huawei Technologies, from 2010 to 2011. He is currently a Senior Engineer with the Hefei Institutes of Physical Science, Chinese Academy of Sciences. His research interests include computer networking, health informatics, blockchain technology, and software architecture.