Journal of Systems Architecture 122 (2022) 102344

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Teegraph: A Blockchain consensus algorithm based on TEE and DAG for data sharing in IoT

Fu Xiang *, Wang Huaimin, Shi Peichang, Zhang Xunhui

College of Computer, National University of Defense Technology, Changsha, 410073, China
Key Laboratory of Software Engineering for Complex System, Changsha, 410073, China

ARTICLE INFO

Keywords:
Blockchain
Consensus algorithm
IoT
Data sharing
TEE
DAG

ABSTRACT

Blockchain offers new ways to the data sharing-based collaboration among IoT devices when a centralized IT infrastructure is unavailable. As one of the critical elements in a Blockchain system, the existing consensus algorithms still have some weaknesses, such as energy-wasting, low throughput, high latency, and increased network communication requirements. In this paper, we focus on designing a highly efficient Blockchain consensus algorithm for data sharing among IoT devices. We present the detailed design of Teegraph, which is a Trusted Execution Environment (TEE) and Directed Acyclic Graph (DAG)-based consensus algorithm. A proof-of-concept implementation of Teegraph is presented. The simulation results demonstrate that TEE usage in Teegraph is more efficient than that of the existing state of the art TEE-based consensus algorithms such as MinBFT and MinZyzzyva. Moreover, Teegraph outperforms Hashgraph, one of the most popular DAG-based consensus algorithms in throughput and latency.

1. Introduction

Data sharing in IoT is the key to IoT devices to collaborate. However, it is not easy for these non-trusting devices to achieve data sharing without a trusted intermediary [1]. Therefore, it is a challenge to share data among the distributed IoT devices when a centralized IT infrastructure is unavailable. According to IBM infographic, Blockchain [2,3], which has been developed for more than ten years, is promised to be a game-changer for IoT [4,5]. Blockchain can offer new ways to the data sharing among IoT devices without setting up a complicated and expensive centralized IT infrastructure [6–8]. To leverage Blockchain into IoT, the first thing to consider is the consensus algorithm [9]. Therefore, we summarize three main requirements for the consensus algorithm used for data sharing in IoT: (1) High-efficient, the consensus algorithm must have a high throughput to process the transactions, and low latency for devices to communicate with each other. (2) Different devices in a device-swarm may be equipped with different hardware for different sub-tasks. For a sub-task, the swarm may be separated, while for another task, these sub-swarms may gather-together. So dynamical changing of the consensus subjects must be archived in the consensus algorithm. (3) The consensus algorithm must be Byzantine Fault-Tolerant. The IoT devices may be attacked by hackers and then have malicious behaviors. Therefore, the consensus algorithm must guarantee that the consensus can still be reached in the presence of these malicious behaviors. According to our analysis, most of the existing consensus algorithms used for crypto-currencies are based on Proof of Work (PoW) or Practical Byzantine Fault Tolerance (PBFT). However, the PoW-based consensus algorithms encounter high computing power costs, long confirmation times, and poor scalability. The PBFT-based consensus algorithms improve the throughput, while the increased requirements for network restricts the system performance in a large-scale environment [10,11]. Although Algorand [12] uses a verifiable random function to select a committee of nodes that participate in a novel Byzantine consensus protocol and solves the scalability problem, this consensus protocol requires a high quality of network connectivity. Therefore, we need to design a more efficient consensus algorithm for IoT Blockchain instead of using the existing ones.

There is an assumption that the initialization of IoT devices is controllable. We can unify these devices with the same hardware or equip them with different hardware according to their functions. Therefore, the Trusted Execution Environment (TEE) can be equipped with them. There have been many studies in the academic field that applies the TEE to distributed consensus algorithms. The TEE is commonly known as an isolated processing environment in which applications can be securely executed irrespective of the rest of the system [13]. TEEs help these algorithms to achieve Byzantine Fault-Tolerance, improve efficiency, and reduce communication overhead [14]. Inspired by these works, researchers have used TEEs to develop more efficient

* Corresponding author at: College of Computer, National University of Defense Technology, Changsha, 410073, China.
E-mail address: fuxiang13@nudt.edu.cn (X. Fu).

https://doi.org/10.1016/j.sysarc.2021.102344
Received 26 August 2021; Received in revised form 9 November 2021; Accepted 13 November 2021
Available online 27 November 2021

Blockchain systems. These studies apply TEEs to the chain-structure Blockchains such as the Teechain [15] and the Proof-of-Luck consensus algorithm [16]. Furthermore, researchers have proposed several consensus algorithms based on Directed Acyclic Graph (DAG) including Hashgraph [17], PHANTOM [18], Conflux [19], Byteball [20] and IOTA [21], which improve the throughput and reduce the latency of processing transactions comparing to the chain-structure Blockchains. However, these Blockchains cannot achieve the dynamic changing of the consensus subjects to adapt to the data sharing-based collaboration for different tasks in IoT. According to our research and analysis, the efficiency can still be improved if we leverage both the TEE and DAG technology in IoT Blockchains. In our recent work [22], which was published as a letter, we proposed a TEE and DAG-based consensus algorithm called Teegraph for IoT Blockchains. Teegraph can help the IoT devices to achieve the data sharing-based collaboration without a centralized third party. The consensus degree of any event/transaction can be shown to all the nodes in real-time. Furthermore, the shared data cannot be tampered once it reaches consensus in Teegraph. All the above features make Teegraph suitable for data sharing in IoT. However, in that letter, we just presented a brief introduction of Teegraph, which may confuse the readers about how Teegraph works. Therefore, in this work, we will describe the detailed design of each key element of Teegraph, including the communication model, the "single-use of self-parent" mechanism, the "dynamic changing of the consensus subjects" mechanism, and the "resource-saving" mechanism. We also prove the safety and liveness of Teegraph. Furthermore, we present several sets of simulation experiments to demonstrate the efficiency of Teegraph.

The major contributions of this paper are:

- **Single-use of self-parent mechanism:** In Teegraph, the usage of TEEs is much simpler and straightforward compared to the existing TEE based consensus algorithms such as MinBFT [14] and A2M [23]. The TEE-based "single-use of self-parent" mechanism reduces the lower bound on the number of total participants from $3f+1$ to $2f+1$, in which f is the maximum number of tolerated Byzantine fault nodes. Furthermore, based on this mechanism, the consensus process for Teegraph is more efficient than that of Hashgraph.
- **Dynamic changing of the consensus subjects mechanism:** Teegraph supports the dynamic changing of the consensus subjects, which means the nodes can join or exit freely without affecting the consensus process.
- **Resource-saving mechanism:** We propose a resource-saving mechanism, which is absent in the original Hashgraph, to reduce the communication overhead and save storage resource when there are no new transactions created. Nodes will stop sending unnecessary empty events after all the non-empty events reaching consensus.

The rest of this paper is organized as follows. Section 2 presents an overview of the related work. Section 3. illustrates the overall system model of Teegraph. We discuss the evaluation framework in Section 4. The simulations and results are discussed in Section 5, and Section 6 concludes this paper.

2. Background and related work

This section summarizes the most relevant works, including the DAG-based consensus algorithms and the TEE-based consensus algorithms.

The DAG-based consensus algorithms: Hashgraph is one of the most popular DAG-based Blockchains. The nodes communicate with each other through a gossip protocol [24]. Moreover, the nodes not only gossip about transactions but also gossip about gossip, which means they also send the gossip history to their neighbors. Every honest node will eventually have the same Hashgraph, which allows the agreement to be achieved through virtual voting: Nodes do not send votes to others over the Internet. Instead, benefit from the gossip about gossip protocol, each node calculates what votes the others would have sent according to the Hashgraph it holds. So nodes can reach the Byzantine agreement for all events locally without any more communication. However, according to our analysis, if a malicious participant creates a fork in Hashgraph, the liveness can never be guaranteed if Hashgraph is deployed as a public Blockchain. Even so, we can use Hashgraph as a consortium Blockchain (The number of the participants for the consensus process is specified, and the identity of the participants are known to each other. We call this version of Hashgraph a consortium version), the liveness problem can be solved because the fork can be found easily. The participants will be kicked out of the consortium if they are caught acting maliciously. Therefore, we implement the consortium version of Hashgraph as the comparison in this paper.

The consensus algorithm of Byteball is also based on DAG. Every node can send a transaction by packing it into a unit, and units are linked to each other such that each unit includes one or more hashes of earlier units. Just like Hashgraph, if unit 1 is an ancestor of unit 2 from node A, it means that node A has confirmed unit 1. There are 12 witnesses who are responsible for the units' finality. The witnesses must create units to confirm other units from the ordinary nodes all the time. Byteball's security relies on the principle that more than half of the witnesses are honest. However, the throughput of Byteball is very low, and the latency depends on the witnesses' unit sending interval. Tangle is a DAG-based consensus algorithm for IOTA. IOTA is also designed for the IoT industry. When a node creates a transaction, it must approve two previous transactions. These approvals are represented by directed edges in DAG, just like Hashgraph and Byteball. If there are conflicting transactions, the nodes need to decide which transaction will become valid by running the Markov Chain Monte Carlo (MCMC) algorithm. However, in the current version of IOTA, there is a fully centralized role called "coordinator" to validate the transactions [25,26]. PHANTOM follows Bitcoin's model in almost every respect, including PoW, computationally bounded attacker, probabilistic security guarantees, etc. The only difference is the data structure: a block references several predecessors rather than a single one. In Conflux, the throughput bottleneck is at the processing capability of individual nodes instead of the consensus algorithm. Similar to Bitcoin, Conflux also operates with the energy-wasting PoW mechanism.

The TEE-based consensus algorithms: A Trusted Execution Environment is a piece of hardware provided by recent commodity CPUs. It is isolated from other parts of the system and can provide security features such as isolated execution and applications' integrity. TEEs are becoming increasingly famous, the most common TEEs include Intel Software Guard Extensions (SGX) [27] and ARM TRustZone [28]. Intel SGX runs on most modern x86 processors, and ARM TrustZone is available on many ARM devices. Recently, many distributed consensus algorithms utilize the power of TEEs to increase efficiency. MinBFT and MinZyzzyva are proposed in [14]. MinBFT is a non-speculative algorithm based on PBFT [29], while MinZyzzyva is a speculative algorithm based on Zyzzyva [30]. In MinBFT and MinZyzzyva, the TEE provides a tamperproof trusted counter service that can produce a signed certificate proving that a certain counter value is uniquely bound to some message. It guarantees that the malicious replica would not make different correct replicas execute different operations as their ith operation. A2M provides the programming abstraction of a trusted log, which leads to protocol designs immune to equivocation—the ability of a faulty host to lie in different ways to different clients or servers [23]. All these algorithms can also be deployed in Blockchain systems. Teechain is a new off-chain payment protocol that utilizes TEEs to perform secure, efficient, and scalable fund transfers on top of a Blockchain, with asynchronous Blockchain access [15]. It brings inspiration for our follow-up work, which is concentrated on the scalability of Blockchains. In paper [16], the authors present how using TEEs

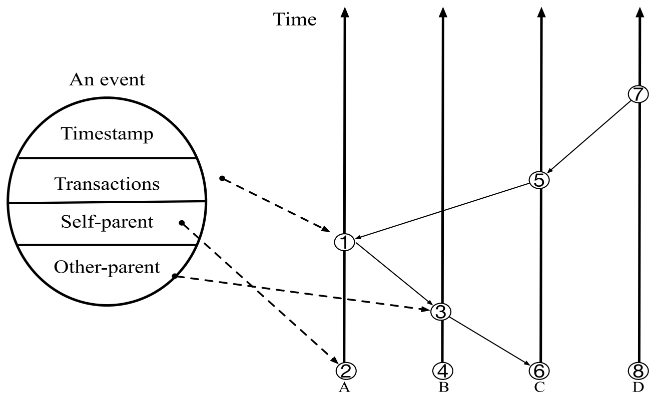

Fig. 1. The Hashgraph.

for existing PoW schemes can make mining equitably distributed by preventing the use of ASICs. They also propose a consensus algorithm called Proof-of-Luck, which uses a TEE platform's random number generator to choose a consensus leader. This algorithm offers low-latency transaction validation.

For Hashgraph, a single-node fork attack needs little cost, while the cost for recovering from a single-node attack is very high. PHANTOM and Conflux still operate with the energy-wasting PoW mechanism. A2M needs much storage for the logs, which brings the obstacle for its large-scale implementation. The usage of TEEs in MinBFT and MinZyzzyva is more complicated than our method. The DAG-based Blockchains can also be improved with the help of TEEs. Furthermore, all the work above cannot supports the dynamic changing of the consensus subjects, which is necessary for IoT scenarios. So in this paper, we design an innovative way to combine TEE and DAG technology for Blockchain consensus algorithm used in IoT scenarios.

3. Design of Teegraph

In Teegraph, each node is responsible for packing its own transactions. A node puts its transactions in an event (the left part of Fig. 1 shows the data structure of an event) and then sends the event to the network by choosing a random neighbor as the destination. The nodes will also receive events from their neighbors. According to the events they receive, nodes can generate DAGs locally. When deciding whether an event reaches consensus or not, each node can calculate what others would vote for an event, so no votes are sending through the network. For example, in Fig. 1, event 2 is an ancestor of event 5 from node C, it means node C has voted "YES" for event 2. Moreover, every node that holds this DAG can calculate how many votes an event gets. That is to say, nodes can reach consensus for all events locally, according to the Teegraph they hold.

3.1. How to generate the graph

We reuse Hashgraph's communication model "gossip about gossip", which is derived from the famous Dynamo proposed by Amazon [31]. Teegraph consists of vertexes and columns. Each column represents a node, and each vertex in the columns represents a gossip event. An event in Teegraph is just like a block in a chain-structure Blockchain. The difference is that in an event, two hash values are linking to its two parent events (There is only one parent for a block in the chain-structure Blockchain). In Teegraph, every vertex, except for the first one in each column, has two downward edges, connecting to the immediately-preceding events called self-parent and other-parent. As shown in Fig. 1, the self-parent of event 1 is event 2, and the other-parent is event 3. The first time node B performing a gossip to node A, it sends all the events it has, including event 3, 4, and 6 to A. Time flows up the graph, so lower vertices represent earlier events in history. The number of nodes is precisely the number of columns in the graph. Every node generates the graph locally by adding events to columns and set edges according to the gossip history. When a node receives events from another node, it adds these events to his own graph. After that, it creates a new event and chooses a neighbor randomly to send events to. Fig. 2 shows the processes of generating the graph for node A, B, and C, respectively:

The initial state. There is only one event for each node: For node A, column A only has event 1, Column B and C are empty; for node B, column B only has event 2, column A and C are empty; and for node C, column C only has event 3, column A and B are empty.

Step 1: Gossip from node A to B. Node A randomly chooses a neighbor (Node B is chosen) to send events to. Before sending events, node A asks node B what node B has in all three columns. Then node A knows that node B only has one event in column B. According to the rule, node A sends all the events it has while node B does not to node B. In this case, node A sends event 1 to node B. After receiving event 1, node B creates a new event (event 4) and sets the other-parent of event 4 as event 1, the self-parent as event 2. In this step, the graphs of node A and C remain the same.

Step 2: Gossip from node B to C. Node B randomly chooses a neighbor (Node C is chosen) to send events to. The same as step 1, node B knows that node C only has one event in column C before sending events. According to the rule, node B sends event 1, 2 and 4 to node C. After receiving these events, node C creates a new event (event 5) and sets the other-parent of event 5 as event 4, the self-parent as event 3. Now node C has event 1∼5. In this step, the graphs of node A and B remain the same.

Step 3: Gossip from node C to A. Now there is a gossip from node C to A. In this case, node C sends event 2, 3, 4 and 5 to node A because at this moment, node A only has event 1. After receiving these events, node A creates a new event (event 6) and sets the other-parent of event 6 as event 5, the self-parent as event 1. In this step, the graphs of node B and C remain the same. If in the next step, node A chooses node C as the event-sending destination, it would only send event 6 because node C already has event 1∼5.

In the above processes, there is only one node sending events in each step. However, nodes can send events in parallel. That is to say, when there is a gossip from node A to B, there may be other gossips from node C to D, node E to F, etc.

3.2. The single-use of self-parent mechanism

To launch a fork attack, a malicious node lies in creating two different events, putting them in the same place on his column (creates two different events and set one same self-parent to them), and then sending them to different neighbors. For most IoT scenarios, the device's initialization is controllable. We can unify these devices with the same hardware or equip them with different hardware according to their functions. So in Teegraph, we leverage the TEE to prevent the fork attack. We design a mechanism called single-use of self-parent, in which TEEs guarantee that every event can be a self-parent only once. Before an event is sent to the network, it must get a TEE's signature, proving that its self-parent is set as a self-parent only once. There are four steps to get the TEE's trusted signature: (1) the node sends event n to the TEE; (2) the TEE compares event n's self-parent hash with the hash of event $n-1$ which has been stored in its memory; (3) if equals, the Tee signs event n and sends it back to the node; (4) the TEE stores the hash of event n to replace the hash of event $n-1$ in its memory. In step 3, if not equal, the TEE dumps event n and stops the process. In a word, if event $n-1$ in a TEE's memory is set as the self-parent of event n, event $n-1$ will be replaced by event n in the TEE's memory immediately. A node can never create two different events with the same self-parent, which means the fork attack can never happen. The algorithm for an event to get a signature from the TEE is Algorithm 1.

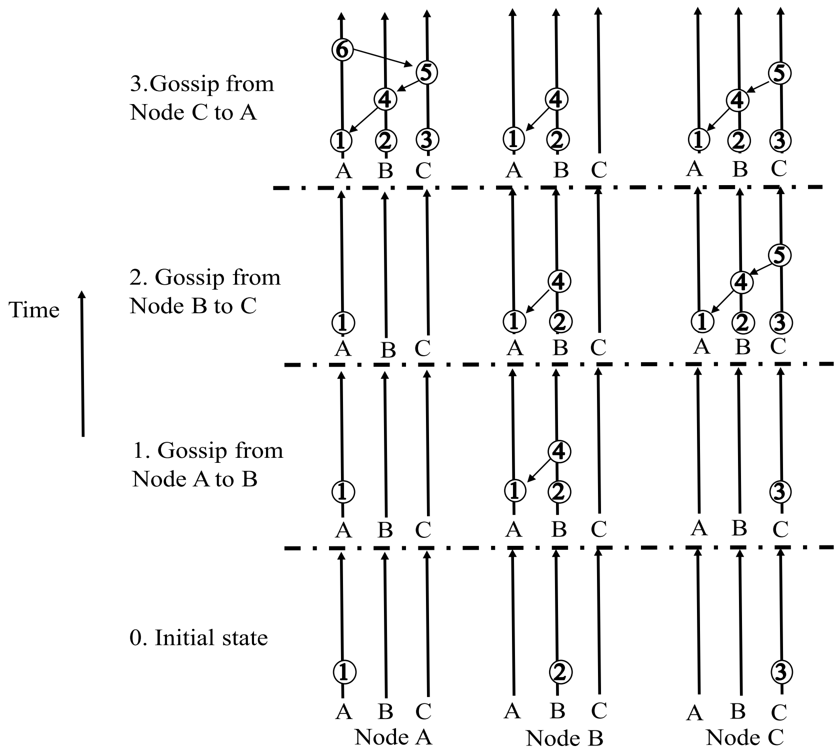

Fig. 2. The process of generating the graph.

Algorithm 1 The algorithm runs on a TEE.

// *temp* is initialized to 0.
Receive an event n from the node
if The self-parent of event n equals *temp* **then**
  Sign event n and sends it back to the node
  Set *temp* to event n
else
  Dump event n
end if

So when receiving an event, the event's validation should add a new item, which is the validation of the TEE's signature. As shown in Fig. 1, at the first time node C receives sync from node A, it receives event 1, 2, 3 and 4 because at that moment, node A has event 1, 2, 3, 4, and 6. In contrast, node C only has event 6 (According to the gossip rule, node A sends all the event it has while node C does not so far. So for every node, an event is received only once). After node C verifies all the received events, it creates a new event 5 and sets its other-parent as event 1, which means node C votes "YES" to event 1, 2, 3, and 4. So if node A creates an event and sets its other-parent as event n from another node, it means that node A has voted "YES" to event n and all the ancestors of event n. So if an event gets more than half of the nodes' votes, which means the event has been verified by half of the nodes, the event reaches consensus. As shown in Fig. 3, all the dark events have reached a consensus because they all have received more than half of the nodes' votes. Just like the famous Crash Fault-Tolerance (CFT) algorithm RAFT [32] and Paxos [33], the nodes cannot send equivocal messages (votes) to others. Therefore, half of the nodes' votes are enough for an event to reach consensus. It is proved in [34,35] that the Byzantine Fault-Tolerance (BFT) problem complexity can be reduced to that of CFT problems if a malicious server cannot lie in different ways to different clients or servers. Moreover, in this case, the lower bound on the number of total participants required to tolerate f faults can be reduced from $3f+1$ to $2f+1$. In Teegraph, with the help of TEEs, a malicious node can never lie to other honest nodes. Hence, the complicated consensus process in Hashgraph is unnecessary, and $2f+1$ nodes are enough to tolerate f malicious nodes. The consensus algorithm runs on a node is Algorithm 2.

Algorithm 2 The consensus algorithm runs on a node.

Run the following two loops in parallel threads
loop
  Send all known events including the event it newly created to a random node
end loop
loop
  Receive events from its neighbor node N
  if All the events are valid **then**
    Create a new event m
    Set the other-parent of event m as the newest event from node N
    Set the self-parent of event m as the last event this node just created
    Send the newly created event m to TEE to get a signature
  else
    Dump all the events received
  end if
  Find new events that reach consensus
end loop

3.3. The dynamic changing of the consensus subjects

Different kinds of devices may form a swarm in some IoT scenarios and work together for a task without a trusted intermediary. Data sharing is required among these non-trusting devices for collaboration. Moreover, the shared data must reach consensus among these devices before the swarm uses it. Most consensus algorithms can achieve this when the consensus subjects are always the same (nodes cannot be replaced, no new nodes join, and no nodes exit). However, there

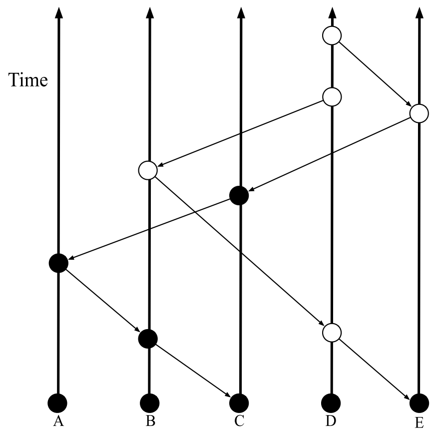

Fig. 3. The consensus process in Teegraph.

are often multiple sub-tasks, requiring the swarm to be separated into several different sub-swarms. For example, in an earthquake relief work, a swarm that is consist of drones and unmanned vehicles needs to be separated for different sub-tasks: the drones will detect the overall environment, and the unmanned vehicles will rescue the discovered victims. After completing their sub-tasks, these sub-swarms may recombine for another new task, as shown in Fig. 4. Therefore, the consensus algorithm for IoT Blockchains needs to have the ability to reach consensus when the consensus subjects change continuously without the support of the trusted intermediary.

When the swarm starts to perform a task, all devices are connected. They can communicate with each other arbitrarily. Therefore, a Teegraph is built. When some devices are separated as sub-swarms to perform sub-tasks, the devices belonging to the same sub-swarm communicate with each other internally. As shown in Fig. 5, sub-swarm A and B are separated from all devices to perform their respective sub-tasks. The events from sub-swarm A (B) reach consensus once they get half "YES" votes from the devices in sub-swarm A (B). After the swarm is recombined, they resume arbitrary communication in the entire network, and the events must get half of "YES" votes from all the devices in the swarm to reach consensus.

3.4. The resource-saving mechanism

In the original Hashgraph, every node needs to create and send events all the time to validate other nodes' events. Even a node has no transaction to send, it must create and send empty events, which contain no transaction. However, when there is no new transaction created and all the existing transactions have reached consensus, these empty events will not contribute to the system anymore but waste the network and storage resources. To solve this problem, we propose a resource-saving mechanism for Teegraph. According to this mechanism, nodes can stop gossiping at the right time: Before a node creates and sends an empty event, it can make a judgment on whether it should do so according to the Teegraph structure it holds. As shown in Fig. 6, the light circles represent empty events, while the dark ones represent the events containing transactions. Event 1 is the latest event that contains transactions, and event 2∼5 have collected enough votes to confirm event 1. When node A creates event 6, it can make sure that all the nodes in the system have confirmed event 1 because event 6 is the offspring of event 2∼5, all of which have collected enough votes for event 1. Moreover, there are not unconfirmed transactions in the system. To reduce network communication costs and save storage resources, node A can stop gossiping after it sends event 6. When another node receives event 6, it can also make sure that all the other nodes have confirmed event 1. So it will stop gossiping. The gossip-based communication will be restarted by any node by creating an event containing new transactions.

3.5. The correctness of Teegraph

In this subsection, we discuss the correctness of Teegraph. As a Byzantine Fault-Tolerant consensus algorithm, Teegraph must guarantee the **Safety** and **Liveness**.

Safety: ***Safety means the consistency of the Blockchain data***. As described in sub Section 3.2, a node can never generate a fork in Teegraph. Although a node can create two conflictive events, there must exist a parent–child relationship between these two events, which means if a node gets the child-event, it must get the parent-event first. According to the mechanism, if the parent-event is invalid, all the child-events of this parent-event are regarded as invalid events. If the child-event is invalid, the nodes dump it. Only the parent-event of these two conflictive events may get more than half of the "YES" votes and reach consensus, proving the Safety in Teegraph.

Liveness: ***Liveness means that valid events created by honest nodes will always reach consensus***. In Teegraph, an event must get the votes from half of the nodes before it reaches consensus. A valid event can always pass the validation and get a "YES" vote when sent to an honest node. If an event can reach the most honest nodes, it can get majority "YES" votes from these honest nodes. The gossip-based dissemination can make the events get spread exponentially fast through the network. In general, it takes O(logN) rounds to reach all nodes, where N is the number of nodes [36,37]. So the gossip protocol guarantees that any event from an honest node can eventually reach all the honest nodes, which proves the Liveness in Teegraph.

4. Evaluation framework

In general, the higher decentralization and security, the lower throughput, such as PoW; or the higher throughput and security, the lower decentralization, such as PBFT. Based on our analysis of the existing Blockchain consensus algorithms and surveys such as [38,39], and [40], we summarize three broad themes of our evaluation framework: effectiveness, security, and decentralization. The algorithms can only meet two aspects and sacrifices the other. It is a paradox of the impossible triangle of Blockchain.

Effectiveness: The effectiveness dimension includes the performance of all aspects of Blockchain, such as throughput, that is, the number of transactions the system can process in a unit of time; the transaction confirmation latency, that is, the time required for a transaction from the generation to the final confirmation; scalability, that is, the number of the nodes that can participate in the consensus; and the resources consumption such as CPU, network bandwidth, memory, computing power, etc. The scalability of Teegraph depends on the gossip protocol. The gossip-based dissemination provides some probabilistic guarantees of message delivery, and the number of times a peer needs to gossip is logarithmic in the size of the system [36,37]. So Teegraph can be deployed to large-scale distributed systems. The throughput and latency will be discussed in sub Section 5.2.

Security: As for Teegraph, we assume that less than 1/2 of the nodes are dis-honest (Teegraph can tolerate more than 1/3 Byzantine nodes because the TEE can prevent the equivocation of nodes). It is also assumed that the asymmetric encryption algorithm and the hash function are secure so that signatures cannot be forged, and hash collisions can never be found. The security dimension refers to the ability of the Blockchain to resist various attacks, such as double spending (attacks the consistency) [41], Denial of Service (DoS), Sybil attacks, selfish mining, etc. The consistency and liveness of Teegraph have been discussed in sub Section 3.5, which indicates that Teegraph can resist double-spending. It is also resilient to DoS attacks because no leader is selected as the accountant. Thus, the attackers cannot decide which node to attack. Sybil attack (A single node creates or manipulates many identities in the network, thus compromising the network) will not succeed because every node must be equipped with a TEE. Selfish mining is only for PoW based Blockchains [42], which

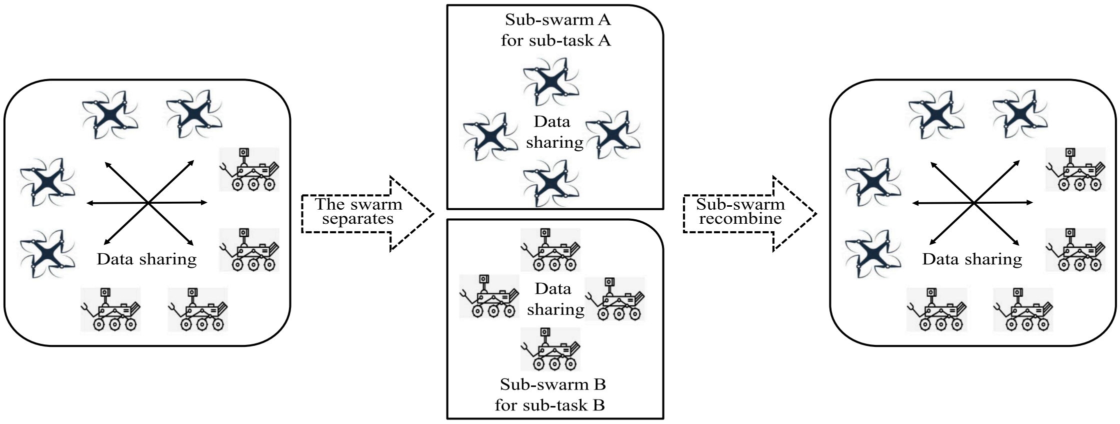

Fig. 4. The swarm separates and sub-swarms recombine.

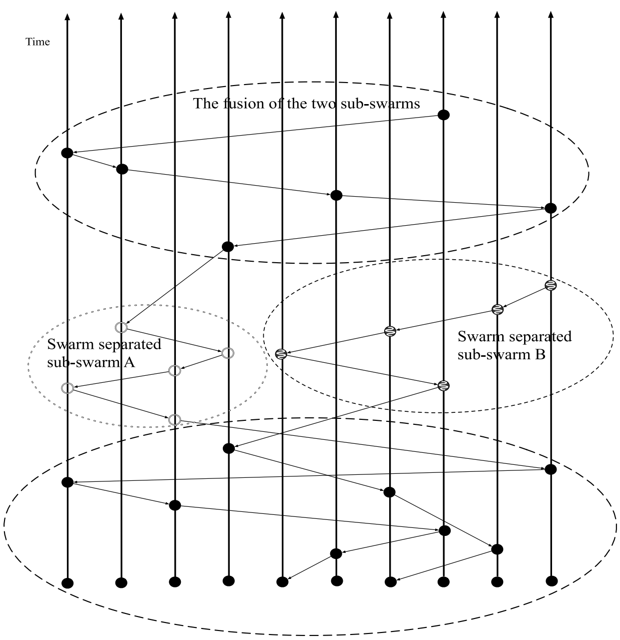

Fig. 5. The Teegraph with the dynamic changing of the consensus subjects.

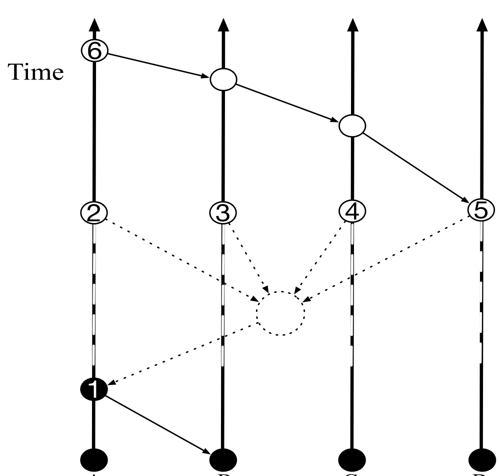

Fig. 6. The resource-saving mechanism.

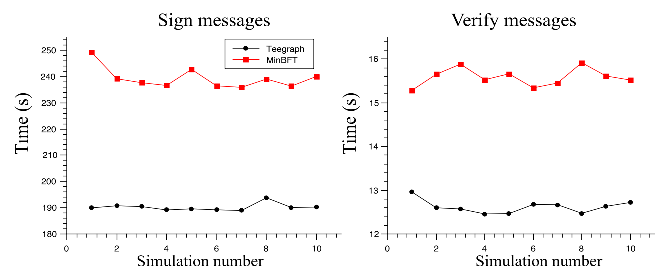

Fig. 7. The comparison of TEE efficiency between Teegraph and MinBFT (each experiment is simulated for 10 times).

affects the fairness of the competition for accounting rights. Therefore, Teegraph is highly secure under our reasonable assumptions.

Decentralization: The decentralization dimension refers to the proportion of the nodes participating in the consensus in the total nodes, whether there are requirements toward participants, and whether the accounting rights distribution is fair, etc. For Teegraph, every node is responsible for his own transactions, so every node is the accountant, which means the accounting rights distribution is fair. Although all the nodes participate in the consensus process, they must trust the TEEs. Hence, to some degree, Teegraph sacrifices decentralization to get high performance and security.

5. Simulations

To demonstrate the efficiency of our algorithm, in this section, we present a proof-of-concept implementation of TEEs within Teegraph.

5.1. Simulation experiments for TEE

In this subsection, we demonstrate the efficiency of TEE in Teegraph. We implement the TEE-based tamperproof trusted counter service in MinBFT as the comparison. In the first experiment, we compare the efficiency of signing a message. Then in the second experiment, we compare the efficiency of verifying a message with the Tee's signature. For the first experiment, we sign 500,000 messages in Teegraph and MinBFT, respectively, and record the time used. For the second experiment, we verify those messages from the first experiment and record the time. Both of the experiments are simulated ten times. The results are shown in Fig. 7. For both signing and verifying messages, Teegraph needs less time than MinBFT. The TEE in MinBFT provides a tamperproof trusted counter service, and every node should store the TEEs' counters of all the other nodes. Moreover, when verifying a message, Teegraph only needs to verify the signature, while MinBFT needs to verify the signature along with the counter's value.

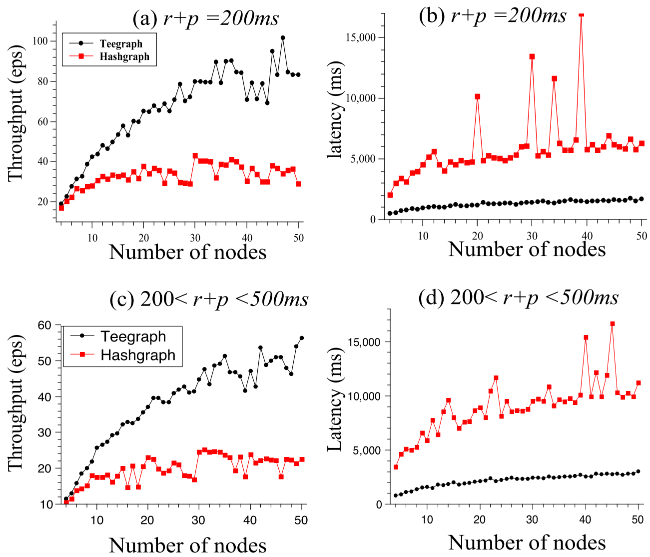

Fig. 8. The comparison of throughput and latency between Teegraph and Hashgraph with different number of nodes.

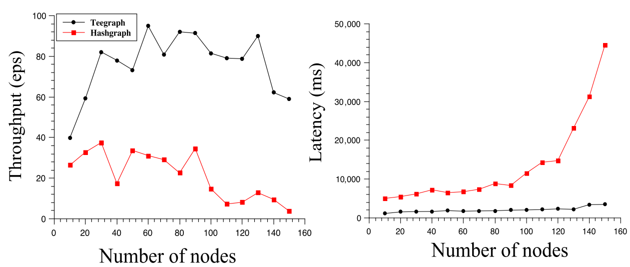

Fig. 9. The comparison of scalability between Teegraph and Hashgraph according to throughput and latency.

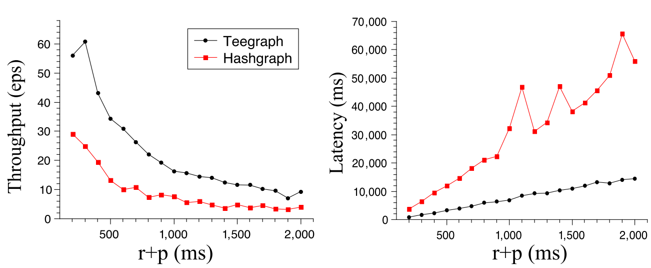

Fig. 10. The comparison of throughput and latency between Teegraph and Hashgraph with different network latency (r+p).

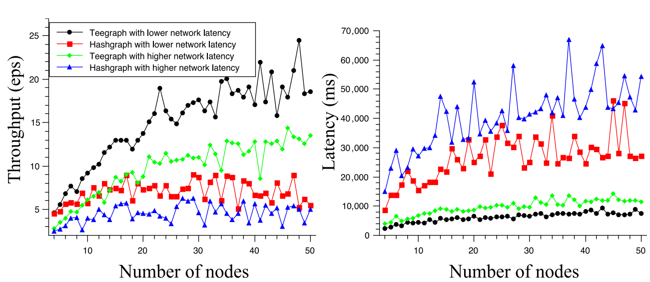

Fig. 11. The comparison of throughput and latency between Teegraph and Hashgraph applying the "fail - skip" strategy.

5.2. Simulation experiments for throughput and latency

In this subsection, each node is simulated as a Java thread. The experiments are running in Eclipse and the nodes (threads) send and receive transactions, validate and confirm events in parallel. We use r to represent the event requesting interval (the nodes generate events periodically and the time interval is r) and p to represent the event propagation time. Then the network delay can be represented by $r+p$. We compare the **throughput** (eps, number of events processed per second) and the **latency** (ltc, the average time for an event to reach consensus, from its generation to confirmation) of Teegraph to those of Hashgraph (the consortium version). The simulations are set in different situations, including different numbers of nodes and different network delay (different event requesting intervals and different event propagation time).

In the first simulation, we study the impact of the number of all nodes. The number of nodes ranges from 4 to 50. We set $r+p=$ 200 ms in the first experiment. Then in a more complex situation, $r+p$ is set to a random value ranging from 200 to 500 ms. The results are shown in Fig. 8. The result suggests that when the number of nodes increases, the latency gets higher. The more nodes, the more events created, but the more votes an event needs to reach consensus. Therefore, the throughput reaches a peak when there are about 30 nodes for Hashgraph (the throughput peak for Teegraph can be found in the next simulation). In a relatively stable network without failure nodes, the latency of Hashgraph has some outliers. That is because for an event to reach consensus, Hashgraph needs three or more rounds of the majority-votes collection (Teegraph only needs one round). Teegraph needs much fewer communication steps than Hashgraph. Therefore, in this simulation, Teegraph outperforms Hashgraph both in throughput and latency. The simulation about the impact of failure nodes can be found in our recent work [22], in which the number of nodes is set to 50 and the number of failure nodes ranges from 1 to 24. Results in that work suggest that the throughput was reduced and latency increased with an increase in the number of failure nodes. Teegraph is not considerably affected when failure nodes increase, and it outperforms Hashgraph. Moreover, when the number of failure nodes is greater than 16, Hashgraph stops working, whereas Teegraph continues to function up to greater than 24 failure nodes.

To analyze the scalability of Teegraph, we add the nodes up to 150. In this simulation, the number of nodes is set to 10, 20, 30, …, 150. These nodes are full nodes that validate every block and transaction by checking them according to the network's consensus rules. Full nodes must also have a copy of the blockchain, so every transaction and block that has ever taken place on the Blockchain must be downloaded. Although there may exist massive IoT devices, most of them are light nodes, which does not need to download all the blocks and transactions. Getting all blocks and transactions is unnecessary for light nodes who just want to send or receive data. They do not care about old transactions. They do not care about other nodes' transactions. They only care for their own transactions. Therefore, light node was invented to save space and computing time. A light node only downloads block headers to validate the authenticity of the transactions. Light nodes cannot join the consensus process. Therefore, the simulation scale (150 full nodes) is reasonable. The results are shown in Fig. 9, which suggest that as the number of nodes increases, the latency is higher. The latency of Teegraph is relatively stable, while when the nodes are more than 120, the latency of Hashgraph has a dramatic increase. The throughput reaches a peak when there are about 60 nodes for Teegraph. Moreover, when the nodes are more than 90, the throughput of Hashgraph has a sharp drop, while Teegraph can still work normally even when the number of nodes is 150 (the throughput of Teegraph is about 60 eps while that of Hashgraph is only about 3 eps). So Teegraph has better scalability than Hashgraph.

Now we study the impact of the network delay (represented by $r+p$) on the throughput and latency. The number of the nodes is set to 50, and we run our simulator for different $r + p$ ranging from 200 ms to $2s$. The results are shown in Fig. 10. The results suggest that the latency is positively related to the network delay. Our algorithm is much less affected than Hashgraph when the network delay increases. Also, our algorithm outperforms Hashgraph in both throughput and latency, no matter what the network delay is.

At last, we perform simulation experiments in more complex situations—the numbers of nodes ranging from 4 to 50, and we apply the "fail - skip" strategy (If a node waits for events for more than 500 ms from a neighbor, it will skip this neighbor and request events from another node). In the first situation, $r + p$ is set to a random value ranging from 100 to 1100 ms, while in the second situation, it is set to a random value ranging from 100 to 1600 ms (the network delay of the second situation is larger than the first one). The results are shown in Fig. 11, which suggest that as the network delay increases, our algorithm is less affected and also outperforms Hashgraph in both throughput and latency.

6. Conclusion

Leveraging Blockchain into IoT offers new ways to the data sharing-based collaboration among IoT devices without setting up a complicated and expensive centralized IT infrastructure. Blockchains can build trust between IoT devices, reduce collusion and tampering risks, and cut down costs by removing overhead associated with middlemen and intermediaries. The Blockchain-IoT combination is powerful and brings significant transformations across many IoT applications. This paper presents the detailed design of a consensus algorithm called Teegraph, which is based on TEE and DAG. Teegraph guarantees that a malicious node can never lie to the others. So the lower bound on the number of total participants required to tolerate f Byzantine Faults is reduced from $3f+1$ to $2f+1$. The consensus degree of every event in Teegraph can be shown to every node in real-time. Therefore, all the IoT devices can decide whether the shared data can be used. Teegraph also supports the dynamical joining or exiting of the IoT devices for different tasks. Moreover, there is a resource-saving mechanism in Teegraph to reduce the communication overhead and save storage when there are no new transactions created. A proof-of-concept implementation of Teegraph is presented in this paper. The simulation results demonstrate that TEE usage in Teegraph is more efficient than that of MinBFT and Minzyzzyva. Moreover, Teegraph outperforms Hashgraph (we implement the consortium version of Hashgraph) both in throughput and latency.

Deploying Teegraph in the real-life scenario is one of our future directions. We will also leverage Teegraph to provide a data, storage, network, and computing resources trading market [43,44] based on smart contract [45,46]. Besides, we plan to focus on cross-chain technology [47–49] in our future work to break the information islands and achieve value transfer among different Blockchains.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61772030, in part by Zhejiang Lab, China (2021PE0AC01) and in part by GF innovative Research Program, China.

References

[1] L.D. Nguyen, I. Leyva-Mayorga, A.N. Lewis, P. Popovski, Modeling and analysis of data trading on blockchain-based market in IoT networks, IEEE Internet Things J. PP (99) (2021) 1–1.
[2] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: IEEE International Congress on Big Data, 2017.
[3] Y. Yuan, F.Y. Wang, Blockchain: The state of the art and future trends, Acta Automat. Sinica (2016).
[4] IBM, Watson IoT and blockchain: Disruptor and game changer, 2018, https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WW912350USEN.
[5] H.-N. Dai, Z. Zheng, Y. Zhang, Blockchain for internet of things: A survey, IEEE Internet Things J. 6 (5) (2019) 8076–8094.
[6] K. Liu, X. Qiu, W. Chen, X. Chen, Z. Zheng, Optimal pricing mechanism for data market in blockchain-enhanced internet of things, IEEE Internet Things J. 6 (6) (2019) 9748–9761.
[7] W. Chen, Y. Chen, X. Chen, Z. Zheng, Toward secure data sharing for the IoV: A quality-driven incentive mechanism with on-chain and off-chain guarantees, IEEE Internet Things J. (2019).
[8] R. Kamal, E.D. Hemdan, N. El-Fishway, A review study on blockchain-based IoT security and forensics, Multimedia Tools Appl. (2021) 1–32.
[9] X. Fu, H. Wang, P. Shi, A survey of blockchain consensus algorithms: mechanism, design and applications, Sci. China Inf. Sci. 64 (2) (2021) 1–15.
[10] V. Gramoli, From blockchain consensus back to Byzantine consensus, Future Gener. Comput. Syst. (2017) S0167739X17320095.
[11] T.T.A. Dinh, W. Ji, C. Gang, L. Rui, B.C. Ooi, K.L. Tan, Blockbench: A framework for analyzing private blockchains, 2017.
[12] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, ACM, 2017, pp. 51–68.
[13] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: what it is, and what it is not, in: 14th IEEE International Conference on Trust, Security and Privacyin Computing and Communications, 2015.
[14] G.S. Veronese, M. Correia, A.N. Bessani, L.C. Lung, P. Verissimo, Efficient Byzantine fault-tolerance, IEEE Trans. Comput. 62 (1) (2013) 16–30.
[15] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, E.G. Sirer, Teechain: Scalable blockchain payments using trusted execution environments, 2017.
[16] M. Milutinovic, W. He, H. Wu, M. Kanwal, Proof of luck: an efficient blockchain consensus protocol, 2017.
[17] L. Baird, The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance, Tech. Rep (2016).
[18] Y. Sompolinsky, A. Zohar, Phantom: A scalable blockdag protocol, 2018, https://eprint.iacr.org/2018/104, Cryptology ePrint Archive, Report 2018/104.
[19] C. Li, P. Li, W. Xu, F. Long, A.C.-c. Yao, Scaling nakamoto consensus to thousands of transactions per second, 2018, arXiv preprint arXiv:1805.03870.
[20] A. Churyumov, Byteball: A decentralized system for storage and transfer of value, 2016, URL https://byteball.org/Byteball.pdf.
[21] S. Popov, The tangle, Cit. on (2016) 131.
[22] X. Fu, H. Wang, P. Shi, X. Ma, X. Zhang, Teegraph: Trusted execution environment and directed acyclic graph-based consensus algorithm for IoT blockchains, Sci. China Inform Sci (2019) http://engine.scichina.com/doi/10.1007/s11432-019-1516-3.
[23] Chun, ByungGon, Maniatis, Petros, Shenker, Scott, Kubiatowicz, John, Attested append-only memory : Making adversaries stick to their word, Acm Sigops Operating Syst Rev 41 (6) (2007) 189–204.
[24] R.V. Renesse, D. Dan, V. Gough, C. Thomas, Efficient reconciliation and flow control for anti-entropy protocols, in: The Workshop on Large-Scale Distributed Systems and MIDDLEWARE, 2008, pp. 1–7.
[25] F.M. Bencic, I.P. Zarko, Distributed ledger technology: Blockchain compared to directed acyclic graph, in: IEEE International Conference on Distributed Computing Systems, 2018, pp. 1569–1570.
[26] P. Schueffel, Alternative Distributed Ledger Technologies Blockchain vs. Tangle vs. Hashgraph - A High-Level Overview and Comparison, Social Science Electronic Publishing, 2018.
[27] V. Karande, E. Bauman, Z. Lin, L. Khan, SGX-Log: Securing System Logs With SGX, in: Acm on Asia Conference on Computer and Communications Security, 2017.
[28] N. Santos, H. Raj, S. Saroiu, A. Wolman, Using ARM trustzone to build a trusted language runtime for mobile applications, 2016.
[29] M. Castro, B. Liskov, Practical Byzantine fault tolerance, in: Symposium on Operating Systems Design and Implementation, 1999, pp. 173–186.
[30] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva:speculative byzantine fault tolerance, in: Acm Sigops Symposium on Operating Systems Principles, 2007.
[31] G. Decandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: amazon's highly available key-value store, Oper. Syst. Rev. (2007).
[32] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: 2014 $\{USENIX\}$ Annual Technical Conference ($\{USENIX\}\{ATC\}$ 14), 2014, pp. 305–319.

[33] L. Lamport, et al., Paxos made simple, ACM Sigact News 32 (4) (2001) 18–25.
[34] L. Lamport, The Byzantine generals problem, Acm Trans. Progr Lang Syst 4 (3) (1982) 382–401.
[35] G. Bracha, An asynchronous (n-1)/3-resilient consensus protocol, in: Proc.Third Ann.Acm Symp.Principles of Distributed Computing, 1984, pp. 154–162.
[36] A.M. Kermarrec, M.V. Steen, Gossiping in distributed systems, Acm Sigops Oper. Syst Rev 41 (5) (2007) 2–7.
[37] P. Eugsterand, R. Guerraoui, A. Kermarrec, L. Massoulie, From epidemics to distributed computing, IEEE Comput 37 (5) (2004) 60–67.
[38] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, G. Danezis, Consensus in the age of blockchains, 2017, arXiv preprint arXiv:1711.03936.
[39] G.-T. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain., J. Inform Proc Syst 14 (1) (2018).
[40] N. Chalaemwongwan, W. Kurutach, State of the art and challenges facing consensus protocols on blockchain, in: 2018 International Conference on Information Networking (ICOIN), IEEE, 2018, pp. 957–962.
[41] G.O. Karame, E. Androulaki, S. Capkun, Double-spending fast payments in bitcoin, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ACM, 2012, pp. 906–917.
[42] I. Eyal, E.G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, Commun. ACM 61 (7) (2018) 95–102.
[43] Z. Li, Z. Yang, S. Xie, Computing resource trading for edge-cloud-assisted internet of things, IEEE Trans. Ind. Inf. 15 (6) (2019) 3661–3669.
[44] T. Alskaif, J.L. Crespo-Vazquez, M. Sekuloski, G.V. Leeuwen, J. Catalao, Blockchain-based fully peer-to-peer energy trading strategies for residential energy systems, IEEE Trans. Ind. Inf. PP (99) (2021) 1–1.
[45] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, M. Imran, An overview on smart contracts: Challenges, advances and platforms, Future Gener. Comput. Syst. 105 (2020) 475–491.
[46] S. Fernández-Vázquez, R. Rosillo, D. Fuente, P. Priore, Blockchain and Smart Contracts: A Revolution, Organizational Engineering in Industry 4.0, 2021.
[47] L. Deng, H. Chen, J. Zeng, L.-J. Zhang, Research on cross-chain technology based on sidechain and hash-locking, in: International Conference on Edge Computing, Springer, 2018, pp. 144–151.
[48] S. Yang, H. Wang, W. Li, W. Liu, X. Fu, CVEM: A cross-chain value exchange mechanism, in: Proceedings of the 2018 International Conference on Cloud Computing and Internet of Things, 2018, pp. 80–85.
[49] I. Williams, Cross-Chain Blockchain Networks, Compatibility Standards, and Interoperability Standards: The Case of European Blockchain Services Infrastructure, Cross-Industry Use of Blockchain Technology and Opportunities for the Future, 2020.

Fu Xiang is a research associate at National University of Defense Technology (NUDT), Changsha, China. He received Ph.D. degree from NUDT in 2020. His research interests are cloud computing, Blockchain and distributed computing technology.

Wang Huaimin is a professor at National University of Defense Technology (NUDT), Changsha, China. He received his Ph.D. degree from NUDT in 1992. He has been awarded the "Chang Jiang Scholars Program" professor by Ministry of Education of China, and the Distinct Young Scholar by the National Natural Science Foundation of China (NSFC), etc. His research interests are middleware, software Agent, trustworthy computing, distributed computing technology.

Shi Peichang is an assistant professor at National University of Defense Technology (NUDT), Changsha, China. He received Ph.D. degree from NUDT in 2012. His research interests are operating systems and distributed computing technology.

Zhang Xunhui is a Ph.D. Candidate at National University of Defense Technology (NUDT), Changsha, China. His research interests are cloud computing, Blockchain and peer-to-peer systems.