SCIENCE CHINA
Information Sciences

• LETTER •

March 2022, Vol. 65 139104:1–139104:3
https://doi.org/10.1007/s11432-019-1516-3

Teegraph: trusted execution environment and directed acyclic graph-based consensus algorithm for IoT blockchains

Xiang FU*, Huaimin WANG, Peichang SHI, Xingkong MA & Xunhui ZHANG

College of Computer, National University of Defense Technology, Changsha 410073*, China*

Received 4 April 2019/Revised 14 May 2019/Accepted 16 July 2019/Published online 21 May 2021

Citation Fu X, Wang H M, Shi P C, et al. Teegraph: trusted execution environment and directed acyclic graph-based consensus algorithm for IoT blockchains. Sci China Inf Sci, 2022, 65(3): 139104, https://doi.org/10.1007/s11432-019-1516-3

Dear editor,

Owing to the recent advancements in the field of Internet of things (IoT), the use of IoT devices has increased in every aspect of human life, such as smart cities, Internet of medical things, and Internet of vehicles [1]. Such devices gather data from the surrounding environment and communicate with each other via the Internet. However, it is difficult for such non-trustable devices to collaborate without a trusted intermediary [2]. According to IBM, blockchains [3,4], which have been developed over the past 10 years, are expected to play a vital role in the field of IoT technology [5]. They can build trust between IoT devices, reduce the risks of collusion and tampering, and reduce costs by eliminating the overheads associated with middlemen and intermediaries. The blockchain-IoT combination is a powerful tool and facilitates significant transformations in several IoT applications. For example, Li et al. [6] proposed the use of a credit-based payment for fast computing resource trading in an edge-assisted blockchain-enabled IoT application; Additionally, a lightweight blockchain-based platform for industrial IoT has been presented to address security, trust, and island connection problems in the process of industrial IoT ecosystem construction [7].

As a key element in blockchain systems, existing consensus algorithms possess some limitations, such as waste of energy, low throughput, high latency, and high network communication requirements [8]. This study focuses on the design of a highly efficient consensus algorithm for IoT blockchains. Herein, Teegraph, which includes a gossip protocol-based message communication mechanism to generate a directed acyclic graph (DAG)-based data structure for a highly efficient consensus process, is proposed. Teegraph uses a TEE-based "single-use of self-parent" mechanism to assure that an IoT device can never equivocate when sending messages. It also assures the dynamic changing of the consensus subject mechanism to deal with the situations wherein device-swarms are separated or gathered together for different tasks. When no new transactions are created, Teegraph acts like a resource-saving mechanism for reducing communication overhead and saves storage space. Herein, a proof-of-concept implementation of Teegraph has been presented. Simulation results demonstrate that Teegraph outperforms Hashgraph [9] from the viewpoint of throughput and latency; herein, we implement the consortium version of Hashgraph.

Teegraph. In Teegraph, there are no accountants for collecting transactions. A node generates transactions, places them in an event, and then sends the event to the network by selecting a random neighbor as the destination. Herein, a node sends to the neighbor all the events that he knows while his neighbor does not. Similar to Hashgraph, every node has a column. According to the received events, nodes can generate DAGs locally, as shown in Figure 1(a)(left). There are four nodes, namely A, B, C, and D. While determining whether an event reaches consensus or not, each node can calculate how other nodes would vote for the given event according to the DAG.

A malicious node launches a fork attack by creating two different events and placing them at the same position in its column, and then sending them to different neighbors, as shown in Figure 1(a)(right). Malicious node B creates two events, events 5 and 5', and they have the same self-parent (event 3). Node B sends them to two different neighbors, i.e., nodes A and C. In Hashgraph, a single-node fork attack requires considerably low cost and cannot be discovered quickly and easily. Once the attack is discovered, it must roll back all the related events; therefore, the cost of recovering from a single-node attack is considerably high. Moreover, if a malicious node continues doing this, events from honest nodes will never reach consensus. Resultantly, the liveness of Hashgraph cannot be guaranteed unless Hashgraph is deployed as a consortium blockchain, wherein the number of participants for the consensus process is specified and the identities of participants are known to each other.

In Teegraph, we take advantage of the TEE for preventing fork attacks. We have designed the "single-use of self-parent" mechanism, wherein TEEs guarantee that each

* Corresponding author (email: fuxiang13@nudt.edu.cn)

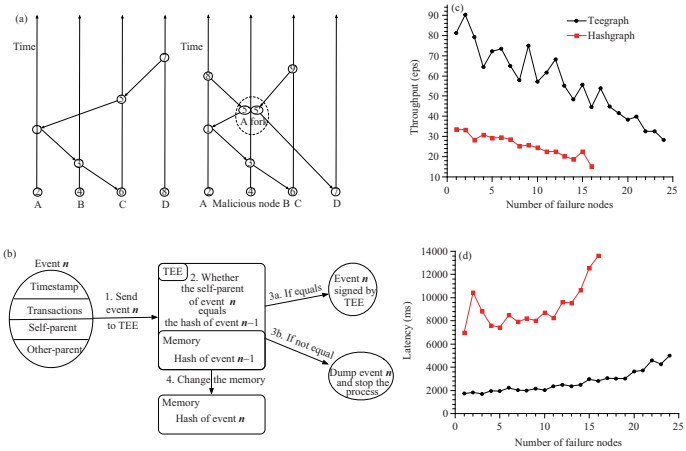

Figure 1 (Color online) (a) Hashgraph (left) and fork attack (right); (b) "single-use of self-parent" mechanism; (c) simulation result of throughput; (d) simulation result of latency.

event can only be a self-parent once. The first event created by a node has no parents; therefore, we set both the parents to zero. As shown in Figure 1(b), before being sent to the network, an event must obtain a signature from the TEE to prove that its self-parent is set as a self-parent only once. All nodes in the system fully trust the TEEs and believe that TEEs do not lie. There are four steps involved in obtaining the TEE's trusted signature: (1) the node sends event n to the TEE; (2) the TEE compares event n's self-parent hash to the hash of event $n-1$ stored in its memory; (3) if both the hashes are equal, the TEE signs event n and sends it back to the node; and (4) the TEE stores the hash of event n to replace the hash of event $n-1$ in its memory. In step 3, if the hashes are not equal, TEE dumps event n and stops the process to wait for the next event. In other words, if event $n-1$ in TEE's memory is set as the self-parent of event n, event $n-1$ will be immediately replaced by event n in the TEE's memory. Therefore, a node can never create two different events with the same self-parent, which means that a fork attack cannot occur. For Hashgraph, before an event reaches consensus, it must obtain votes from greater than 2/3 of all nodes for no less than three rounds. On the contrary, Teegraph only requires votes from greater than 1/2 of all nodes in a single round because the TEE prevents fork attacks.

In some IoT scenarios, different types of devices may form a swarm and work together to accomplish a task without a trusted intermediary. Data sharing between these non-trustable devices is required for such collaboration, and the shared data must reach consensus among these devices before the swarm uses the data. Most consensus algorithms can achieve this when the consensus subjects are always the same, e.g., when nodes are not replaced, no new nodes join and no nodes exit. However, often there are multiple sub-tasks, which require the swarm to be separated as several different sub-swarms. After completing their sub-tasks, these sub-swarms may recombine for another new task. Therefore, when consensus subjects change continuously without the support of a trusted intermediary, the consensus algorithm for IoT blockchains must have the ability to reach consensus. In Teegraph, any node can trigger the swarm separation or sub-swarm' combinations by creating a special event containing a node-list of the new swarm. If a node agrees with this proposal, it will store this event and only communicate with nodes in the new node-list; otherwise, it will dump the given event. When this special event reaches half of the nodes in the list, the consensus subjects' changing becomes effective. Then, an event reaches consensus when it obtains more than one-half of the votes from the node-list.

The original Hashgraph requires each node to create and send events all the time. Even a node that does not have a transaction to send must create and send empty events containing zero transactions. However, when no new transactions are created and all transactions in Hashgraph have reached consensus, empty events do not contribute to the system, i.e., such empty events waste the network and storage resources. In Teegraph, to address this limitation, we propose a resource-saving mechanism that allows nodes to stop gossiping at an appropriate time. Before a given node creates and sends an empty event, the node can evaluate whether it should do so according to the Teegraph structure it holds.

Simulation. To demonstrate the efficiency of the proposed algorithm, we present a proof-of-concept implementation of the TEE within Teegraph. In this simulation, there were a total of 50 nodes, and we simulated different numbers of failure nodes ranging from 1 to 24 ($24 < 50/2$). Each node was simulated as a Java thread. Herein, r and p denote event request interval (rather than sending events to neighbors, each node requests events from its neighbors) and event propagation time, respectively. Therefore, network delay is expressed as $r + p$. Herein, $(r + p)$ was set to 200 ms during the simulation. We compared the throughput (eps: the number of events handled per second) and latency (ltc: the average time for an event to reach consensus) of Teegraph to those of Hashgraph (consortium version). Results are shown in Figures 1(c) and (d). Results 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 ($16 < 50/3 < 17$), Hashgraph stops working, whereas Teegraph continues to function up to greater than 24 failure nodes.

Conclusion. TEE and DAG are two innovative technologies used to build highly efficient blockchains. Herein, we propose an efficient Byzantine fault-tolerant consensus algorithm, Teegraph, for IoT device collaboration. We use the gossip protocol for message communication and present

a TEE-based "single-use of self-parent" mechanism to guarantee that a malicious node can never launch a fork attack. Then, we design a DAG-based consensus process that can support dynamic changes to the consensus subjects. We also design a resource-saving mechanism, which is not available in Hashgraph, to reduce communication overhead and storage requirements when no new transactions are created. A proof-of-concept implementation of Teegraph is also presented, and the simulation results demonstrate that the proposed Teegraph outperforms Hashgraph from the viewpoint of throughput and latency. In the future, we plan to focus on distributed consensus on resource-limited IoT devices, blockchain security, and decentralized schemes for IoT.

Acknowledgements The work was supported by National Key R&D Program of China (Grant No. 2016YFB- 1000100), National Natural Science Foundation of China (Grant No. 61772030), and GF Innovative Research Program.

References

1 Banerjee M, Lee J, Choo K K R. A blockchain future for Internet of things security: a position paper. Digit Commun Netw, 2018, 4: 149–160

2 Christidis K, Devetsikiotis M. Blockchains and smart contracts for the Internet of things. IEEE Access, 2016, 4: 2292–2303

3 Zheng Z B, Xie S A, Dai H N, et al. An overview of blockchain technology: architecture, consensus, and future trends. In: Proceedings of IEEE International Congress on Big Data, 2017

4 Yuan Y, Wang F Y. Blockchain: the state of the art and future trends. Acta Autom Sin, 2016, 42: 481–494

5 IBM. IBM Watson IoT platform. 2019. https://www.ibm.com/us-en/marketplace/internet-of-things-cloud ?mhsrc=ibmsearch_a&mhq=blockchain%20iot

6 Li Z N, Yang Z Y, Xie S L, et al. Credit-based payments for fast computing resource trading in edge-assisted Internet of things. IEEE Int Thing J, 2019, 6: 6606–6617

7 Bai L, Hu M, Liu M, et al. BPIIoT: a light-weighted blockchain-based platform for industrial IoT. IEEE Access, 2019, 7: 58381–58393

8 Bach L, Mihaljevic B, Zagar M. Comparative analysis of blockchain consensus algorithms. In: Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018. 1545–1550

9 Baird L. The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. Technical Report, 2016