Tornado: Enabling Blockchain in Heterogeneous Internet of Things through A Space-Structured Approach

Yinqiu Liu, Kun Wang, *Senior Member, IEEE,* Kai Qian, Miao Du, and Song Guo, *Senior Member, IEEE*

Abstract—With the widespread applications of Internet of Things (IoT), e.g., smart city, business, healthcare, etc., the security of data and devices becomes a major concern. Although blockchain can effectively enhance the network security and achieve fault tolerance, the huge resource consumption and limited performance of data processing restrict its deployments in IoT scenarios. Observing the heterogeneity and resource constraints, we intend to make blockchain accommodate both wimpy and brawny IoT devices. In this paper, we present Tornado, a high-performance blockchain system based on space-structured ledger and corresponding algorithms, to enable blockchain in IoT. Specifically, we first design a space-structured chain architecture with novel data structures for promoting the network scalability. To address the huge heterogeneity of IoT, a novel consensus mechanism named Collaborative-Proof of Work is developed. Moreover, we propose the Space-Structured Greedy Heaviest-Observed Sub-Tree (S^2GHOST) protocol for improving the resource efficiency of IoT devices. Additionally, a Dynamic Weight Assignment mechanism in S^2GHOST contributes to reflect the trustworthiness of data and devices. Extensive experiments demonstrate that Tornado can achieve a maximum throughput of 3464.76 transactions per second. The optimizations of propagation latency and resource efficiency are 68.14% and 30.56%, respectively.

Index Terms—Internet of Things (IoT), Blockchain, Scalability, Heterogeneity, Consensus Mechanism.

1 Introduction

As an emerging technology, the Internet of Things (IoT) is reshaping various sectors, such as manufacturing [1], healthcare [2], transportation [3], etc. With the advances of wireless sensor networks (WSN) and embedded platforms, IoT accommodates numerous smart devices, including sensors, robotics, and vehicles. In this case, massive data can be captured and analyzed to support decision making, thereby sparking academic innovations and leading the era of smart industry (Industry 4.0) [4].

Simultaneously, the in-depth applications pose high requirements for IoT, especially the concerns about trustworthiness and security [5]. In IoT scenarios, they mainly refer to the dependability of data and devices, and the capability of fault tolerance. Generally, IoT systems employ servers or clouds for storage, authorization, and certification. Such centralized strategies are highly vulnerable to Denial of Service, Sybil attacks or tampering [6, 7]. If certain vital facilities become malicious, the whole IoT network may crash. Moreover, the anonymity and heterogeneity of IoT devices cause difficulties for fault tolerance [8]. With the field develops, the lack of security becomes a bottleneck restricting the further applications of IoT.

- *Y. Liu, K. Qian, and M. Du are with the College of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China (e-mail: yinqiuliu@foxmail.com, isqiankai@gmail.com, and dumiao0118@163.com).*
- *K. Wang is with the Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, 90095, USA (e-mail: wangk@ucla.edu).*
- *S. Guo is with The Department of Computing, The Hong Kong Polytechnic University, Hong kong, 999077, China (e-mail: song.guo@polyu.edu.hk).*
- *Corresponding authors: Kun Wang and Song Guo.*

Fortunately, deploying blockchain in IoT to resolve such concerns receives widespread research interests [9-11]. As an append-only database stored in decentralized peer-to-peer (P2P) network, blockchain guarantees data integrity, traceability, and non-tampering [12]. Furthermore, through eliminating the reliance on certificate authorities (CA), security and fault tolerance are implemented in non-trusted environments. Although blockchain can effectively enable trustworthiness in IoT, such accomplishments require tremendous resources for support. Meanwhile, the performance of blockchain is limited due to distributed manners. For instance, Bitcoin, the first-known blockchain design, can only confirm 7 transactions per second (TPS), while the confirmation latency is approximately 60 minutes and the electric energy consumption is awful [13]. In this case, deploying blockchain in IoT systems will decrease the performance of most power-constrained devices and the quality of service (QoS) also drops.

To address the above issues, various strategies are proposed, including computation offloading [14, 15] and authoritative entities [16, 17]. Nevertheless, these proposals cannot diminish all the constraints of blockchain in IoT. We observe that the combination of blockchain and IoT faces two daunting challenges. 1) Since IoT involves severs, sensors, embedded and edge platforms, etc., the diversity of devices leads to significant hardware heterogeneity. Similarly, the network topologies, infrastructures, and protocols are also heterogeneous [18, 19]. Heterogeneity causes two obstacles. First, in IoT scenarios, only several brawny devices can support blockchain. The reduced peer number results in higher vulnerability to attacks. To accommodate wimpy devices, some previous work introduced authoritative entities, e.g., managers [6]. However, such settings undermine the

decentralization of blockchain. Second, the heterogeneity will affect blockchain's performance. For example, the peers owning limited bandwidth might fail to synchronize the latest ledger state, leading blockchain into partition. Further, the devices with poor computing power cannot join in ledger maintenance. In the long term, the Matthew effect [10] will be intensive, thereby weakening the willingness of low-end participants. Therefore, IoT-oriented blockchain should be open for both wimpy and brawny devices. 2) The overall performance of IoT devices is insufficient, especially compared with the dedicated equipment used in Bitcoin. Furthermore, IoT networks exhibit dynamics since most battery-powered sensors will frequently switch modes and then join or leave the P2P networks [20]. In this case, the scalability of blockchain systems is also required.

Based on these observations, we propose a novel high-performance blockchain system with space-structured ledger architecture called Tornado to enable blockchain in IoT. Specifically, Tornado accommodates IoT devices of different configurations and improves the scalability in heterogeneous and resource-constrained IoT networks. Meanwhile, the degree of network decentralization and security remains at a high level. The main contributions of our paper are listed as follows.

- To the best of our knowledge, we are the first to propose the concept of space-structured ledger. Through novel data structures, this advanced architecture endows outstanding scalability for Tornado.
- To overcome the heterogeneity, we design an anti-heterogeneity and high-performance consensus algorithm named Collaborative Proof of Work (Co-PoW) based on the collaborations among IoT devices. Co-PoW introduces parallel workflows, which support both wimpy and brawny devices and effectively improve the network throughput.
- To exploit the resource efficiency, we present a novel Space-Structured Greedy Heaviest-Observed Sub-Tree (S^2GHOST) protocol. During S^2GHOST, the Dynamic Weight Assignment (DWA) contributes to measure the trustworthiness of data and devices.
- We also develop a prototype of Tornado. Experiments in a heterogeneous P2P network of 50 peers illustrate that Tornado can achieve a peak throughput of 3464.76 TPS. Moreover, the performance of power efficiency, propagation latency, and scalability also gets significantly optimized.

The rest of this paper is organized as follows. We describe the related work and our motivations in Section 2. In Section 3, we demonstrate the network model and data structures of Tornado. Then, the proposed algorithms are described in Section 4, including Co-PoW and S^2GHOST. In Section 5, we analyze the resistance of Tornado against attacks. Section 6 illustrates the implantations and evaluations of Tornado. The future work is discussed in Section 7. Section 8 concludes this paper.

2 Related Work and Motivations

2.1 Features of IoT and Our Motivations

Recently, IoT has attracted interests from both industrial and academic fields, such as big data [21], energy management [22, 23], business [5, 24], etc. We summarize that IoT scenarios display the following three features.

- ***Heterogeneity***: IoT devices mainly consist of servers, management facilities, and diverse embedded sensors [25]. Among them, sensors are generally wimpy devices, with limited computing power. However, servers possess relatively higher performance, i.e., brawny devices. Moreover, the compositions of communication strategy in IoT are complex, including Wi-Fi, Zigbee, Bluetooth, etc [26]. Therefore, IoT scenarios exhibit significant heterogeneity [18].
- ***Resource constraints***: In general, IoT devices are apparently resource-constrained, especially battery-powered ones. In most cases, even the computation and storage resources or power supplement of centralized servers in management hubs cannot keep up with professional blockchain equipment [10]. Hence, traditional computation-intensive blockchain designs cannot be supported by IoT devices.
- ***Dynamics***: The on-off mode switching of power-constrained devices brings dynamics for IoT networks [19, 20]. In addition, IoT devices show great mobility in certain scenarios, for instance, smart cars in vehicle-to-vehicle networks. Consequently, they will frequently join or leave the network or move positions. The fluctuations of network scale raise strict demands for scalability and stability.

Motivated by these challenges, we intend to develop a blockchain system that could effectively address heterogeneity, resource-intensive workflows, and poor performance, thereby enabling blockchain in IoT scenarios.

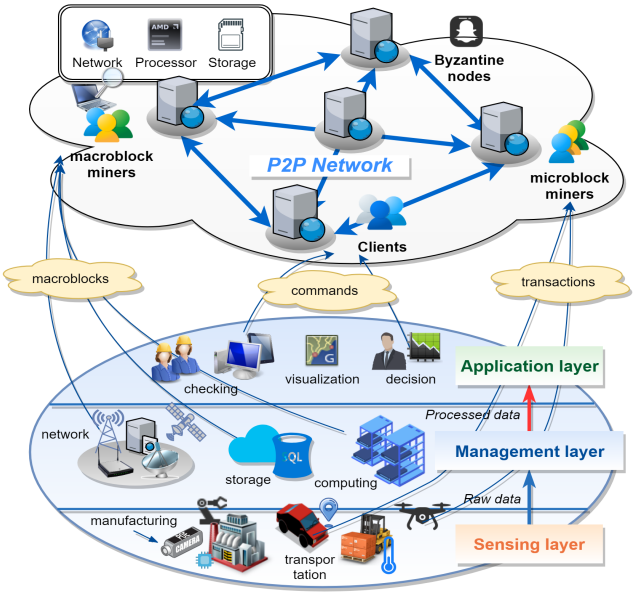


Fig. 1: Network model of Tornado.

2.2 Applications and Optimizations of Blockchain-Enabled IoT

From the first-known system named Bitcoin, blockchain technology has been applied in various fields to enable security and trustworthiness. For instance, H. Li *et al.* [27]

developed a trust-enhanced information-centric networking architecture based on blockchain for content delivery. D. Liu *et al.* [28] proposed an anonymous reputation system, which exploited blockchain to increase the data transparency and reliability. For IoT scenarios, authors of [7] and [15] developed two blockchain platforms for the scurity in wireless IoT and vehicular clouds, respectively. Moreover, J. Huang *et al.* [6] presented the blockchain system with credit-based consensus mechanism for IoT to tackle single point failure and malicious attacks.

To improve the efficiency of IoT-oriented blockchain, the above work organized data into a directed acyclic graph (DAG). Considering the constrained resources of IoT devices, A. Dorri *et al.* [15] introduced a centralized entity called block manager (BM). Similar to management hubs, BM provides a shared key among home devices and controls all incoming and outgoing transactions requests. Additionally, S. Biswas *et al.* [17] organized IoT devices into groups and then employed local CAs to mitigate computation and storage consumption. In these proposals, most low-end devices are not accessible to P2P network, thereby damaging the decentralization of blockchain. From another perspective, M. Liu *et al.* [29] proposed a deep learning algorithm to help IoT applications select block producers and adjust parameters, thereby increasing the performance. W. Chen *et al.* [14] proposed the multi-hop cooperative and distributed computation offloading solution to optimize computation-intensive blockchains for IoT. Nonetheless, these proposals ignore the obstacles from heterogeneity, thus, the high performance cannot be held simultaneously on wimpy and brawny devices. Additionally, the ledger architectures of existing proposals remain linearity or DAG, which cannot exploit the resources of IoT networks.

3 System Model

In this section, we take the Tornado-assisted smart factory scenario as a case study to demonstrate the topology of Tornado networks. Then, we give the detailed descriptions of essential data structures proposed by Tornado.

3.1 Model of Tornado-Assisted IoT Networks

Fig. 1 illustrates the network model of a smart factory, which employs Tornado to enable security. From back-end to front-end, this IoT network owns three layers, i.e., sensing layer, management layer, and application layer. The sensing layer is composed of diverse smart devices, e.g., drones, robotics, and vehicles. Since sensors merely act as data collectors, their processors are low-end micro-computers with weak computing power. Meanwhile, they generally communicate in WSNs and the storage is based on SD cards. Note that there also remains other embedded sensors with even less volume of resources. Such devices are usually battery-powered and suffer from frequent mode switches. In contrast, the management layer monitors numerous sensors, processes massive data, and supports upper-layer industrial applications. Therefore, this layer tends to apply high-end servers and cloud services, resulting in strong computing power, sufficient network traffic, and storage space. In the application layer, administrators can check real-time data via edge platforms then make decisions about manufacturing,

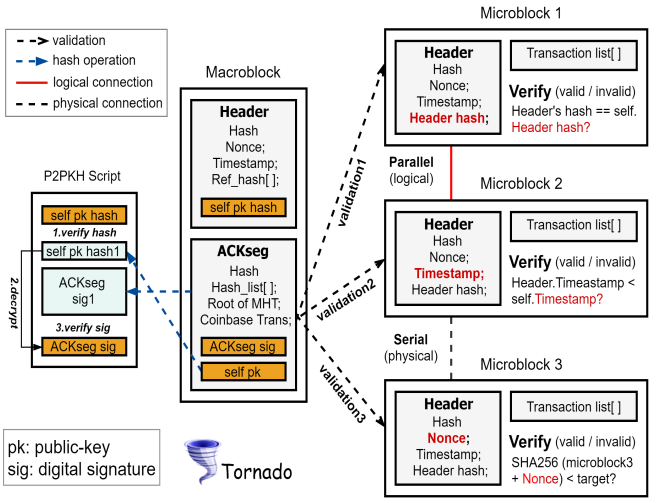


Fig. 2: Data structures of macroblock and microblock.

transportation or marketing. Clearly, the huge heterogeneity sets obstacles for enabling blockchain in IoT scenarios since most devices cannot meet the resource requirements.

To this end, we assign three roles in Tornado for different levels of devices, i.e., macroblock miners, microblock miners, and clients. Naturally, two kinds of miners are responsible for creating corresponding blocks. In addition, they interconnect with each other, process requests from outside clients, and thus form a P2P network. To handle these tasks, the minimum hardware requirements contain one available network interface, one micro processor, and some storage space, as shown in Fig. 1. Given the lightweight workflow, any qualified devices can at least serve as the microblock miners. However, for high-end nodes, being macroblock miners can exploit their capability and enhance their influence in Tornado network. The parallel workflows of miners will be elaborated in Section 4. Finally, edge platforms in the application layer are termed as clients. Instead of saving the complete ledger, clients submit transactions following users' commands and access the blockchain through connected miners. Note that the Byzantine nodes in Fig. 1 refer to malicious miners which might launch attacks, e.g., selfish mining, to destroy Tornado.

Since the function partitions accommodate both wimpy and brawny devices, the heterogeneity of IoT gets effectively addressed. The correspondence between devices in different layers and their suggested roles are plotted in Fig. 1. Different from previous work, the roles of IoT devices under Tornado are not fixed. Peers are encouraged to switch their tasks based on hardware configurations or real-time workloads. As for adapting dynamics, Tornado is designed as a public-chain that adopts the permissionless identity management strategy. In this case, devices are allowed to arbitrarily join or leave the P2P network without verifying their authorities.

3.2 Data Structure

3.2.1 *Macroblocks and Microblocks*

In order to maximize the efficiency of both macroblock and microblock miners, we design dedicated data structures, namely macroblocks and microblocks. As shown in Fig. 2,

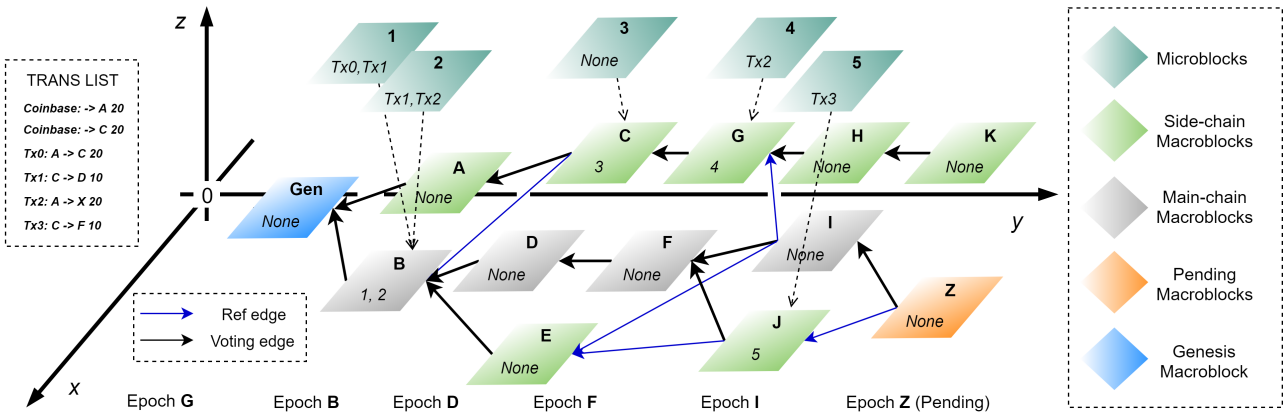


Fig. 3: Space-structured chain architecture of Tornado.

TABLE 1: The notations in Tornado

Notations	Meanings
$\langle \alpha, \beta \rangle$	*The current role of localhost*
(γ, δ)	*The (public, private) key pair of nodes*
φ	*The serial number of the next instruction*
Gen	*The genesis block of Tornado*
$\mathbb{M}$	*The hash set of all main-chain macroblocks*
$\mathbb{B}$	*The hash set of all validated macroblocks*
$\mathbb{E}$	*The set of edges* $\in \mathbb{G}$
$\mathbb{S}$	*The set of all receiving microblocks*
$\mathbb{L}$	*The set of references* $\in \mathbb{G}$
η	*The total number of vertexes* $\in \mathbb{G}$
H^{ack}	*The ACKseg of block header* H
Δ_m	*The creation interval of macroblocks*
Δ_{mi}	*The creation interval of microblocks*
$\Gamma_g(V)$	*The weight of vertex* V *under GHOST*
$\Gamma_s(V)$	*The DW of vertex* V *under* S^2GHOST
χ^{bin}	*The binary type of object* χ
Tips($\mathbb{G}$)	*Return the set of vertexes with in-degree 0*
Child($\mathbb{G}$, V)	*Return the set of vertexes pointing to* V
Subtree($\mathbb{G}$, V)	*Recursively call Child*$(\mathbb{G}, V_i)$ *on child* V_i
Reachable($\mathbb{G}$, V)	*Return* $\mathbb{R}$:= *the set of reachable vertexes*
UnReachable($\mathbb{G}$, V)	*Return the set of unreachable vertexes:* $\mathbb{B} - \mathbb{R}$

all the block creations in Tornado need to solve Co-PoW puzzles by finding one valid nonce, whose process will be shown in Section 4. However, due to the harder Co-PoW puzzles, proposing macroblocks is more time-consuming and resource-intensive than microblocks and is usually executed by power-sufficient devices. Macroblock consists of a header and an acknowledgment segment (ACKseg). The former carries the control information of ledger extension, including hash, Ref_hash that points to the previous macroblocks, pending Co-PoW nonce, Unix timestamp, and SHA-256 digest of miner's public key ($self\_pk\_hash$). The latter contains the confirmation messages of one macroblock, including a hash list of microblocks, coinbase transaction that rewards itself, root of Merkel Hash Tree (MHT), and Base58-encoded public key ($self\_pk$).

The validity of macroblock headers is protected by Co-PoW nonce. Provided that attackers attempt to tamper the header content, they should recalculate a qualified nonce, which is equal to recreate a macroblock. Particularly, ACKsegs will undergo a P2PKH (Pay to Public Key Hash) styled script [30] for digital signature to ensure the reliability and integrity. In Fig. 2, macroblock miners sign the newly created ACKseg using own private key δ, then broadcast ACKseg over the P2P network. For other miners that receive this ACKseg, they will first extract $self\_pk$ then compare the hash of $self\_pk$ ($self\_pk\_hash1$) with the previously received $self\_pk\_hash$. If $self\_pk\_hash1 = self\_pk\_hash$, receivers can determine that the received macroblock header and ACKseg belong to the same sender. After that, receivers decrypt the signature in ACKseg ($ACKseg\_sig$) and sign ACKseg for acquiring $ACKseg\_sig1$. If $ACKseg\_sig = ACKseg\_sig1$ is also satisfied, peers can believe that the received ACKSeg suffers no tampering.

Based on macroblocks, we divide the running time of Tornado into sequential *epochs*, defined as the interval between two adjacent macroblock creations. The macroblock miner who successfully solves Co-PoW puzzle becomes the leader of the current *epoch*, others remain followers. Each node possesses an individual state vector $\vec{\sigma} := \langle \vec{\kappa} := \langle \alpha, \beta \rangle, \mathbb{P} := (\gamma, \delta), \mathbb{G} := (\eta, \text{Gen}, \mathbb{M}, \mathbb{B}, \mathbb{E}, \mathbb{S}, \mathbb{L}), \varphi \rangle$ in local. Here both α and β in $\vec{\kappa}$ are integer and represent the current role of localhost, i.e., $\alpha = 0 \rightarrow$ microblock miner; $\alpha = 1, \beta = 1 \rightarrow$ leader; $\alpha = 1, \beta = 0 \rightarrow$ follower; $other\ values \rightarrow$ clients. Given that peers in P2P networks experience frequent peer updates and message exchanges, $\vec{\kappa}$ is used when establishing connections. For example, client A will request the $\vec{\kappa}$ of peer B before appending it to the neighbor list. If the received $\vec{\kappa}$ is $\langle 2, 1 \rangle$, indicating that B also belongs to clients, A can avoid querying data form B since B saves no ledger as well. $\mathbb{P}$ indicates the (public, private) key pair specified by nodes. φ marks the serial number of the next instruction. Moreover, $\mathbb{G}$ stores the elements of local ledger architecture. The notations used in this paper are shown in TABLE 1.

Microblocks mainly package pending transactions and basic block parameters, such as hash, Unix timestamp, Co-PoW nonce, etc. Besides, each microblock should be assigned to only one leader via $Header\_hash$. Considering that transactions account for the vast majority of microblocks' capacity, microblock miners could control the propagation latency and traffic usage by adjusting the number of packaged transactions. As shown in Fig. 2, any pending microblock can get confirmed by its targeted leader only if it passes three validations: *1) identity validation*: verify whether the microblock is sent to the correct leader; *2) timestamp validation*: verify whether the microblock is created in the current *epoch*; *3) Co-PoW validation*: verify whether the Co-PoW nonce of pending microblock is qualified.

3.2.2 Space-Structured Chain Architecture

Currently, relentless efforts have been invested in chain structure advances, such as the evolutions from linearity/tree [31], to DAG [32], and to committee-based sharding [33]. Enlightened from previous work, Tornado proposes a novel space-structured ledger, as shown in Fig. 3. Such architecture exploits the computation and network resources of IoT network, where control information propagates faster than transactions. In this way, the devices with limited bandwidth can rapidly synchronize the vital elements (i.e., $\vec{\kappa}$, η, $\mathbb{M}$, $\mathbb{B}$, $\mathbb{E}$, and $\mathbb{L}$) of $\vec{\sigma}$. Thus, the instability and poor throughput attributed to network heterogeneity gets optimized. Moreover, such architecture effectively accommodates Co-PoW. We will use a space-rectangular coordinate system *O-XYZ* to demonstrate our chain architecture.

1) Observation of XoY Plane: Backtracking to genesis, the subsequent macroblocks reference multiple parents and then form a DAG structure. The goal of constructing DAG foundation is to improve the computing power efficiency of IoT devices. Furthermore, DAG better reflects the real-time state of IoT network than linearity, including the network throughput, workloads, and trustworthiness, thereby contributing to DWA in S^{2}GHOST. In the P2P network of Tornado, the DAG foundation of local ledger is composed of the following four elements:

- ***Vertex***: Macroblocks are termed as vertexes. Tips refer to the vertexes whose in-degree is 0.
- ***Edge***: Edge indicates the reference relationship between two vertexes. For example, leader fills *Ref_hash* via the hashes of all tips, after creating pending Z.
- ***Voting edge***: Among edges, the voting edges point to the tip of the current main-chain (the gray chain in Fig. 3). Since leaders vote for the main-chain through S^{2}GHOST protocol, voting edges can be regarded as affirmative votes.
- ***Ref edge***: Ref edges point to the remaining tips (the green macroblocks on all side-chain in Fig. 3) and indicate a chronological relation. We note that the definitions of main-chain and side-chain will be demonstrated in Section 4.

In local ledger $\mathbb{G} := (\eta, \mathsf{Gen}, \mathbb{M}, \mathbb{B}, \mathbb{E}, \mathbb{S}, \mathbb{L})$, η represents the total number of vertexes. Gen means the genesis block of Tornado. $\mathbb{B}$ refers to the set containing the hash values of all validated macroblocks. Similarly, $\mathbb{M}$ refers to the subset of $\mathbb{B}$ whose elements belong to the main-chain. $\mathbb{E}$ is denoted as the edge set. An element $e_{\langle \mathsf{B},\mathsf{D}\rangle} \in \mathbb{E}$, in the form of $\langle$ B, D $\rangle$, means the voting edge from B to D.

2) Observation of XoZ Plane: Based on the DAG foundation composed of macroblocks, microblocks (the blocks denoted by: 1-5 in Fig. 3) that carry pending transactions embody the space-structured features. In Tornado, two kinds of blocks are created in parallel. Each macroblock header $H \in \mathbb{B}$ is pointed by numerous validated microblocks following the instruction of corresponding ACKseg H^{ack}. The references between macroblocks and microblocks are denoted by $\mathbb{L} := \{(i, H.hash) \,|\, i \in \mathbb{B},\ H \in \mathbb{S},\ i \in H^{ack}.Hash\_list\}$, where $\mathbb{S}$ preserves all receiving microblocks.

From the perspective of transaction processing, all microblocks based on the same macroblock are mutually independent (e.g., 1 and 2 that point to B). Contributed to such independence, Tornado eliminates the power competition among resource-constrained devices (microblock miners), where numerous peers could propose microblocks and record real-world transactions in parallel. Consequently, the throughput of IoT networks will increase as more peers participate, endowing Tornado an outstanding scalability. Furthermore, the maximum amount of microblocks referencing one macroblock is not fixed but varying with the fluctuations of device number and workload. In this case, sensors are allowed to propose microblocks when detecting fresh data and sleep when the IoT network is idle. This strategy is vital for battery-powered sensors. As to macroblock miners, validating microblocks also bring benefits because the appointments of microblock miners reflect their trustworthiness and contribute to improve the dynamic weight (DW) of the proposed macroblocks.

4 Algorithm Design

In this section, we give the detailed demonstrations of proposed algorithms that support the space-structured ledger of Tornado, including Co-PoW and S^{2}GHOST.

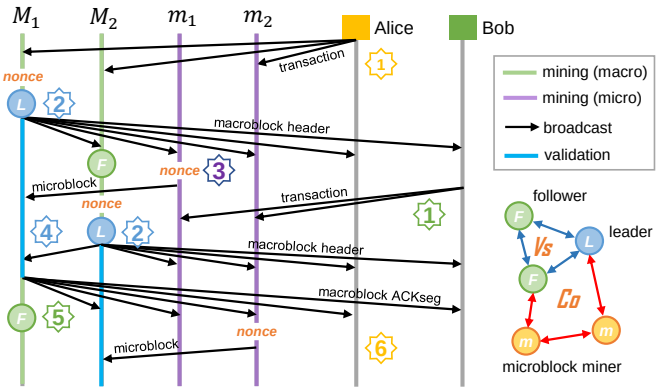


Fig. 4: Parallel workflows of Co-PoW.

4.1 Anti-Heterogeneity Consensus Algorithm for IoT

As mentioned before, Tornado adopts permissionless strategies to match the dynamics of IoT networks. For most permissionless blockchains, Proof-of-Work (PoW) is widely-adopted consensus mechanism since it ensures high-level network security and realizes the fault tolerance. However, PoW views the workload of calculating hash collisions as the standard for verifying miners (the peers that participant in consensus), resulting in massive computing power consumption [34, 35]. Under PoW, miners take the binary content of the previous block and a random nonce as inputs then execute hash operations, called mining. The process of mining can be described as:

$$Find\ nonce\ s.t.\ \ H(nonce\,||\,H(block^{bin})) < target, \quad (1)$$

where H means the hash function; $target$ is pre-defined. Only when the miner finds a qualified nonce can it obtain the right for block creation. Note that mining in PoW follows a competitive manner, i.e., as long as one miner finishes Eq. (1), all remainders update $block^{bin}$ then restart mining.

Obviously, such mining competition is not suitable for IoT scenarios, where devices exhibit huge heterogeneity and

resource constraints. Therefore, the throughput of deploying traditional PoW-based blockchain systems in IoT stays extremely poor. Motivated by this, we propose Co-PoW, which develops the differentiated mining difficulty and parallel workflows for different roles. Through the collaborative ledger maintenance from macroblock and microblock miners, the network throughput can be dramatically promoted. In addition, the hardware heterogeneity of IoT network gets effectively addressed.

4.1.1 Differentiated mining difficulty

Difficulty is negatively correlated to the $target$. Since macroblocks carry the whole control information, their creations require higher difficulty to defend Sybil attacks and achieve the Byzantine Fault Tolerance (BFT) [36]. Conversely, microblocks merely record data flows via transactions, which will be checked by leaders through strict validations. Thus, the workload requirements of Co-PoW for microblock miners are greatly lower, which enable that wimpy devices could solve Eq. (1) within 1s. In the P2P network of Tornado, macroblock miners are generally operated by personal computers or servers with strong computing power. Meanwhile, numerous resource-constrained devices can execute the workflow of microblock miners. Following this strategy, Tornado effectively accommodates both wimpy and brawny devices. Owing to differentiated mining difficulty, resource-strained peers can directly access to blockchain, thereby enhancing the network decentralization. Moreover, Co-PoW retains the security advantage of PoW, since any block creation still requires hash workload for proof.

4.1.2 Parallel workflows

The parallel workflows of macroblock and microblock miners under Co-PoW are shown as **Algorithm 1**. Furthermore, we take the Tornado network with 2 macroblock miners (M_1 and M_2), 2 microblock miners (m_1 and m_2), and 2 clients (Alice and Bob) as an example for better displaying the parallel workflows. Suppose that Alice submits one transaction T to its connected peers, in the next part, we will demonstrate how peers perform their own workflow and confirm T based on **Algorithm** 1 and Fig. 4.

1) *Macroblock miner*: After macroblock miner M_1 finds one qualified Co-PoW nonce, localhost will call the On_Macroblock_Header_Mined ($nonce$) to create a pending macroblock header H_p. This function mainly consists of two key steps: 1) append the last element of $\mathbb{M}$ to $Hp.Ref\_hash$, thereby confirming the voting edge, and 2) call Tips ($\mathbb{G}$) to find all tips in $\mathbb{G}$ and append their hashes to $Hp.Ref\_hash$ as ref edges. Then, H_p is broadcast by M_1 (step 2).

If H_p gets confirmed in the P2P network, M_1 will become current leader and starts its own *epoch* (step 4). Other macroblock miners remain followers and keep mining. During each *epoch*, leader is responsible for validating microblocks transferred from collaborating microblock miners (step 4). So, the microblock that carries T will reach M_1 and be validated. When receiving the next macroblock header H_{p1}, which means the current *epoch* is ended, function On_Macroblock_Header_Received (H_{p1}^{bin}) will be triggered. In this case, M_1 collects all microblocks verified within its *epoch* then fulfills the $Hash\_list$ of H_{p1}^{ack}. Apart from broadcasting ACKseg, its own role switches from leader to follower (step 5). Finally, Alice is informed that T gets packed in macroblocks (step 6) and M_1 restarts mining process. Overall, the relationship between the leader and followers is still based on PoW-styled computing power competitions, as these help improve network security and realize fault tolerance.

2) *Microblock miner*: For microblock miners, the updates of macroblock header and ACKseg only affect data synchronizations, while not terminating their running tasks. As the mining processes complete Co-PoW calculations, m_1 creates and broadcasts the microblock B_p via the function On_Microblock_Mined ($nonce$) (step 3). Other microblock miners, which receive the pending B_p, will call the function On_Microblock_Received (B_p^{bin}). This function validates all transactions encapsulated in B_p. Note that microblock miners are supported to offload duplicate transactions from local transaction pool, which can mitigate transaction overlaps and help to improve the data quality. Similar to various microblocks that point to the same macroblock, the workflows of different microblock miners following one leader are also independent. Therefore, numerous IoT devices can collect nearby data in parallel and handle the workload of the entire network by collaborative manners.

Algorithm 1 Workflows of Collaborative Proof-of-Work

procedure WORKFLOW-OF-MACROBLOCK-MINERS
  server.serve_forever( )
  if On_Macroblock_Header_Mined(*nonce*) **then**
    Create and broadcast macroblock header H_p
    $\langle \alpha, \beta \rangle \leftarrow \langle 1, 1 \rangle$
    $(\eta, \mathbb{B}, \mathbb{E}) \leftarrow (\eta^*, \mathbb{B}^*, \mathbb{E}^*)$
    Validate receiving microblocks
  end if
  if On_Macroblock_Header_Received(H_{p1}^{bin}) **then**
    if $\langle \alpha, \beta \rangle = \langle 1, 1 \rangle$ **then**
      Create and broadcast the $ACKseg$ of self proposed H_p
      $\langle \alpha, \beta \rangle \leftarrow \langle 1, 0 \rangle$
      Start new mining process
    else
      $(\eta, \mathbb{B}, \mathbb{E}) \leftarrow (\eta^*, \mathbb{B}^*, \mathbb{E}^*)$
    end if
  end if
  if On_ACKseg_Received($ACKseg^{bin}$) **then**
    Validate and append the microblocks in $ACKseg.Hash\_list$ to local ledger
  end if
  if On_Microblock_Received(M_r^{bin}) **then**
    Save M_r^{bin} in local memory pool
  end if
end procedure

procedure WORKFLOW-OF-MICROBLOCK-MINERS
  server.serve_forever( )
  if On_Microblock_Mined(*nonce*) **then**
    Create and broadcast microblock B_p
    $(\mathbb{S}, \mathbb{L}) \leftarrow (\mathbb{S}^*, \mathbb{L}^*)$
    Start new mining process
  end if
  if On_Macroblock_Header_Received(H_r^{bin}) **then**
    $(\eta, \mathbb{B}, \mathbb{E}) \leftarrow (\eta^*, \mathbb{B}^*, \mathbb{E}^*)$
  end if
  if On_ACKseg_Received($ACKseg^{bin}$) **then**
    Validate and append the microblocks in $ACKseg.Hash\_list$ to local ledger
  end if
  if On_Microblock_Received(B_r^{bin}) **then**
    Remove duplicate transactions from trans pool (optional)
  end if
end procedure

As shown in Fig. 4, the relationship between microblock and macroblock miners is also collaborative since they cooperate in creating one ledger entry. Benefit from two kinds of collaborations, the hardware heterogeneity of IoT is effectively solved. With Co-PoW, even power-constrained peers can join the consensus and maintain ledgers, their contribution mainly comes from appointing leaders, which is vital for DWA and S^{2}GHOST. To overcome network heterogeneity, the asynchronous broadcast strategy enables peers to receive and transfer macroblocks, only occupying hundred of bytes, in real time. Since the control information and transactions are partitioned, a slight latency of microblock synchronizations is acceptable. In this way, wireless devices with constrained network resources can keep up with the chain extensions and appoint their microblocks to the latest leader. In terms of storage, miners are allowed to use private clouds for saving massive transactions. Nevertheless, viewd from the network layer, Tornado employs no centralized servers for providing storage services like traditional IoT systems. As a result, Tornado is unaffected by single point data failure and still holds decentralization.

4.2 Optimizations of Resource Efficiency

The situation where two macroblocks reference the same parent is a common concern in IoT scenarios due to high workload and limited network performance. The conflicts of chains are called forks, where the pending chain obtaining the most confirmations becomes the main-chain (gray chain in Fig. 3), and others remain side-chain (all green chains in Fig. 3). Well-known blockchain designs, such as Nakamoto Consensus of Bitcoin [37] and GHOST of Ethereum [31], only acknowledge the workload of the main-chain but discard side-chain. However, side-chain also contains considerable valid transactions and qualified consensus workloads. Especially for IoT devices, whose computing, storage, and network resources are insufficient, the drop of resource efficiency caused by simply denying side-chain is unacceptable. To this end, we propose an improved ordering protocol named S^{2}GHOST for Tornado. Adopting DWA mechanism, S^{2}GHOST can order the DAG foundation into a linear chain, thereby processing the transactions on all forks. Furthermore, DWA is tightly relevant to the trustworthiness of macroblocks, leading the macroblock order in Tornado is capable of maximizing the protection of trusted data and improving the data quality.

As described in **Algorithm 2**, S^{2}GHOST is divided into two procedures, i.e., DWA and Greed-Based Traversal.

1) ***Dynamic Weight Assignment (DWA)***: For $\mathbb{G}$ in Fig. 3, $\mathbb{E}$ can be divided into $\mathbb{E}_V$ and $\mathbb{E}_R$, which represent the set of voting edges and ref edges, respectively. In traditional GHOST, only $\mathbb{E}_V$ will be maintained, making ledger structure degenerate into a parental tree. Assuming that each vertex shares fixed weight ω, for vertex V, its weight in GHOST is recursively defined as:

$$\Gamma_g(V) = \omega \ + \sum_{i \in \mathsf{Child(V)}} \Gamma_g(i), \quad \Gamma_g(\mathsf{tip}) = \omega. \tag{2}$$

However, S^{2}GHOST divides the DW of a given vertex into three major parts, i.e., Cardinal Value (CV), Data Validity (DV) and Degree of Contact (DoC).

Algorithm 2 S^{2}GHOST Ordering Protocol

Input: $\mathbb{G}, \Theta, V$(starting point);
Output: $\mathbb{G}^*$ (updated local ledger state)

procedure GREED-BASED-TRAVERSAL($\mathbb{G}, V$)
  if Child $(\mathbb{G}, V) = \varnothing$ **then**
    return V
  else
    for all $v_i \in$ Child $(\mathbb{G}, V)$ **do**
      for all $v_{ij} \in$ Subtree $(\mathbb{G}, v_i)$ **do**
        DW_i += DWA $(\mathbb{G}, v_{ij})$
      end for
    end for
    $V \leftarrow v_i$ with the maximum DW
    $\mathbb{M} = \mathbb{M} \cup \{V.hash\}$
    return Greed-Based-Traversal $(\mathbb{G}, V)$
  end if
end procedure

procedure DYNAMIC-WEIGHT-ASSIGNMENT($\mathbb{G}, V$)
  Initialize $CV = 0, DV = 0, QoC_V = 0, DW = 0$
  $CV = \omega\,(\Delta_m + |C_V| \cdot \Delta_{mi})$
  for all $i \in C_V$ **do**
    DV += Num (valid trans in microblock$_i$)
  end for
  $DoC = |C_V| + |$Reachable$(\mathbb{G}, V)|$
  Calculate DW using Eq. (4)
  return DW
end procedure

CV: CVs of macroblocks and microblocks are defined as $\omega \cdot \Delta_m$ and $\omega \cdot \Delta_{mi}$, respectively. Parameter $\Delta_m\,(\Delta_{mi})$ represents the average creation interval of macro (micro) block within the latest 2000 *epochs*. No matter what kind of blocks, CV is positively proportional to the required time. There are two functions of CV adjustment: 1) during the power-sufficient *epochs* where blocks are created rapidly, lower CV can reduce the influence of power fluctuations on ordering; 2) peers will be encouraged to propose blocks in idle time due to higher CV, which is conducive to adjusting network workloads. Like GHOST, ω is a fixed value being set before initializing Tornado network.

DV: DV is defined as the total number of valid transactions carried by each macroblock. Suppose that the transactions packaged by V constitute a complete set, then DV indicates the size of the remaining set after subtracting all duplicate and conflicting elements, denoted by $|\mathbb{T}_V|$. For instance, in Fig. 3, $|\mathbb{T}_\mathsf{B}| = 3$, $\mathbb{T}_\mathsf{B}$ is {Tx0, Tx1, Tx2}, where Tx1 appears twice. Nonetheless, only the first Tx1 in microblock 1 will be included in $\mathbb{T}_\mathsf{B}$. Tx2 is a duplicate transaction in $\mathbb{T}_\mathsf{B}$ and $\mathbb{T}_\mathsf{G}$, S^{2}GHOST only accepts the first occurrence (i.e., in B), while discards Tx2 when calculating $|\mathbb{T}_\mathsf{G}|$. Similarly, Tx3 conflicts with Tx1, indicating the same input is spent multiple times, in which case S^{2}GHOST is subject to the first one, (i.e., Tx1). By taking DV into consideration, Tornado acquires a greater chance of selecting the macroblocks that carry the most valid transactions to the main-chain, thereby maximizing the network throughput.

DoC: The design of DoC stems from PHANTOM [38], where the concept of *k-cluster* was presented to measure the

inter-connectivity of a block in DAG structure. *k-cluster* is based on a theorem that forks produced by malicious peers will have lower inter-connectivity than those from honest ones, i.e., there will be more vertexes in Unreachable($\mathbb{G}$, V).

Since Tornado organizes macroblocks to the DAG foundation, it shares an analysis model similar to PHANTOM. We redefine inter-connectivity in the space-structured ledger of Tornado. For V, DoC_V is defined as:

$$DoC_V = |C_V| + |\mathsf{Reachable}(\mathbb{G}, V)|, \tag{3}$$

where $|C_V|$ indicates the number of microblocks verified by V. By taking $|C_V|$ into account, numerous microblock miners can join the consensus, thereby endowing Tornado higher decentralization level than previous work. Additionally, $|\mathsf{Reachable}(\mathbb{G}, V)|$ represents the number of vertexes $\in \mathbb{B}$ that V can reach or be connected through edges. For C in Fig. 3, the $|C_{\mathsf{C}}|$ and $|\mathsf{Reachable}(\mathbb{G}, \mathsf{C})|$ are 1 (microblock 3) and 8 (Gen, A, B, G, H, I, K, Z), respectively.

DoC accounts for a large proportion of DW. For forks mined by honest nodes, DoC is not sensitive to their locations in the DAG foundation. However, for the forks created by byzantine nodes, DoC can effectively reveal their attacking attempts. This is because the attackers can only tamper one transaction by creating an alternative chain full of conflicting transactions in secret. At one certain time, attackers will broadcast the whole chain to P2P network. If the alternative chain could attain higher DW than the current main-chain, conflicting transactions manipulated by attackers would replace normal ones, known as selfish mining [39]. For traditional NKC and GHOST, such problem is serious. However, in S^{2}GHOST, since alternative chains remain private until attackers broadcast them, the microblocks created by honest microblock miners cannot appoint to malicious leaders. Likewise, the macroblocks created by honest leaders cannot reference malicious macroblocks. Consequently, malicious chains usually suffer form extremely low DoC. In DWA, the DW is defined as:

$$\begin{aligned}\Gamma_s(V) = \omega\left(\Delta_m + |C_V| \cdot \Delta_{mi}\right) + \\ \frac{|\mathbb{T}_V| \cdot DoC_V}{\Theta} + \sum\nolimits_{i \in \mathsf{Child(V)}} \Gamma_s(i),\end{aligned} \tag{4}$$

where Θ represents the average network throughput over the latest 2000 *epochs*. Note that adjusting Δ and Θ every 2000 *epochs* is to keep their accuracy for reflecting the real-time block creation interval and throughput of Tornado network. As to the adjustment rate, it can be set according to the practical deployment cases. The higher the rate is, the more accurate the DWA is. However, if Δ and Θ change too frequently, peers might enter in partitions then disturb the system. In the Bitcoin network with over 10000 globally-distributed nodes, parameter adjustments happen every 14 days. Provided that macroblock miners create one macroblock per minute, Tornado network will update Δ and Θ every 1.38 days. Since such a frequency is suitable for IoT networks whose size is relatively small, "2000 *epochs*" represents our suggested value.

Up to now, we elaborate the definitions and functions of three components that make up DW. DWA views DW as the metric for measuring the priority of macroblocks, which is the premise of ordering local ledger. In the next part, we will introduce the process of proposed Greed-based Traversal to complete the S^{2}GHOST.

2) ***Greed-based Traversal***: Just like its name, S^{2}GHOST falls in the category of greedy algorithms. Following a BFS (Breadth First Search) method, S^{2}GHOST traverses $\mathbb{G}$ level by level. At each level, it executes DWA for every vertex then selects the one with the greatest DW as the starting point v for entering the next level, i.e.:

$$\text{Select } v \in \mathsf{Child}(V) \quad s.t., \max_{v \in \mathsf{Child}(V)}(\Gamma_s(v)). \tag{5}$$

After reaching one tip of $\mathbb{G}$, the set of sequentially selected v constructs the main-chain of Tornado (line 11 of **Algorithm 2**). For those remaining elements $\in \mathbb{B}$, they are assigned to the nearest main-chain macroblocks, e.g., in Fig. 3, C is assigned to B, J is assigned to F, etc. If more than one macroblocks belong to the same *epoch*, the relative order is positively correlated to their DW. In this way, S^{2}GHOST successfully orders the space-structured ledger into a linear chain, where DV maximizes the network throughput.

Apart from the increased performance of transaction processing, DWA offers help for measuring trustworthiness. The trustworthiness of data (mainly refers to macroblocks) is reflected by DW itself since DoC is sensitive towards attacks, as we proved earlier. Moreover, DV further expands the gap of DW between the macroblocks created by honest and those from Byzantine nodes. This is because clients tend to submit transactions to the miners that often lead the *epoch*, for faster confirmations. Clearly, when preparing selfish mining attacks, the secret behaviors of attackers are the opposite with clients' preferences. Recall that S^{2}GHOST orders the ledger based on DW, the final order is consistent with data trustworthiness where the macroblocks created by potential attacks will be arranged behind.

As to IoT devices, for simplicity, we measure their trustworthiness from their proposals. In detail, if one certain peer continues to create macroblocks whose DW is obviously lower than most counterparties in the same *epoch*, we judge that this device is controlled by attackers. Then, Tornado can dilute the power of this peer, such as commanding honest nodes refuse to broadcast its proposals and appending this peer in the black list. To enhance the effectiveness of DW regarding the device trustworthiness measurements, incentive mechanisms can be introduced. However, since this paper mainly focus on overcoming the heterogeneity and enabling blockchain systems in IoT scenarios, the part of incentives is left as a future work.

5 Security Analysis

Security is an important concern in evaluating blockchain-assisted IoT. In this Section, we analyze the resistance of Tornado when facing various kinds of attacks.

5.1 Attacks against IoT

1) Data tampering and single point failure: In traditional IoT systems, every node only stores its relevant data, i.e., each file is maintained by only related nodes. Moreover, centralized CAs are employed for issuing digital certificates. Therefore, IoT systems are highly vulnerable to single point failure. If the key devices storing vital data are intruded

TABLE 2: The configurations of heterogeneous testbed.

Role	configuration (CPU, Cores, RAM, Bandwidth)	Number	Hashrate	Location
Macroblock miner	*Intel Xeon E5 (3.2Ghz), 16 cores, 32GB, 5Mbps*	*3*	*1.4237 MHPS*	*Shanghai*
	Intel Skylake (2.5Ghz), 8 cores, 16GB, 2Mbps	*5*	*1.1823 MHPS*	*Nanjing*
Microblock miner	*ARM Cortex-A7 (1.2Ghz), 4 cores, 2GB, 2Mbps*	*20*	*0.1124 MHPS*	*Shanghai*
	ARM Cortex-A72 (1.5Ghz), 4 cores, 4GB, 1Mbps	*22*	*0.1882 MHPS*	*Nanjing*
Wallet software	*Simulated by 50 × automatic transaction generator.*			

Note: *Hashrate is evaluated by Python's hashlib.sha256() library, whose unit is mega hashes per second (MHPS).*

by attackers, the whole system might crash. In contrast, Tornado requires each miner hold one copy of blockchain. So, attackers can only modify the local storage of all peers for data tampering, which is nearly impossible. In addition, the reliance on CAs is eliminated, thereby realizing the fault tolerance and diminishing the single point failure.

2) Sybil attack: In Sybil attacks, malicious nodes illegally present themselves with multiple identities, and thus sending forged messages to victims. To defend sybil attacks, Tornado retains the PoW-styled consensus mechanism. Since the creations of blocks, especially macroblocks still demand hash workload for proof. In this cases, attackers cannot arbitrarily forge blocks unless they solve Co-PoW puzzles. Furthermore, DW can assist peers in measuring the trustworthiness of devices, which is conductive to detecting the potential Sybil nodes.

5.2 Attacks against Blockchain

1) Selfish mining: Selfish mining is a major threat to blockchain. In normal cases, each transaction input can only be spent once. However, through selfish mining, attackers attempt to overtake normal transactions with conflicting ones then retrieve their payments, called double-spending. The process of launching selfish mining is elaborated in Section 4.2. As described above, both DoC and DV contribute to defend against selfish mining. Recall that the order of suspicious macroblocks with lower DW is behind honest ones after executing S^2GHOST, the malicious transactions have little chance to overtake normal transactions. Since Tornado confirms the first one when handling conflicting transactions, attackers' proposals will be ignored. In conclusion, compared with standard GHOST, Tornado apparently mitigates the effects from selfish mining attacks.

2) Lazy leader: This is a special strategy attacking the blockchain that employs leaders. For lazy leaders, they would deliberately violate the normal workflows to disturb the network. Such methods include referencing old vertexes rather than tips, disordering received microblocks, and delaying the broadcast of ACKseg, all of which have negative impact on system stability. Fortunately, the proposed mechanisms help Tornado reduce the risk of lazy leaders. For instance, the DAG foundation allows macroblock miners to skip the lazy leader and enter in the next *epoch*. Similarly, microblock miners can easily change leaders due to the simple Co-PoW puzzles. Further, the timestamps provide a deterministic order of microblocks. Overall, the power of lazy leaders gets successfully constrained in Tornado.

6 Implementation and Evaluations

1) Implementation: To evaluate the performance of our proposals, we develop a prototype of Tornado in Python 3 and C++. Three components make up the implementation of Tornado, i.e., macroblock miners, microblock miners, and clients. Moreover, we develop two blockchain systems driven by standard Nakamoto Consensus (NKC) and GHOST to illustrate the advantages of Tornado.

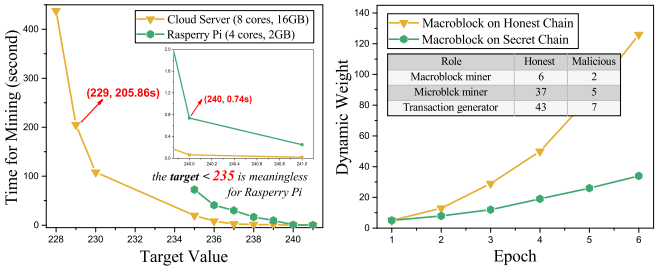
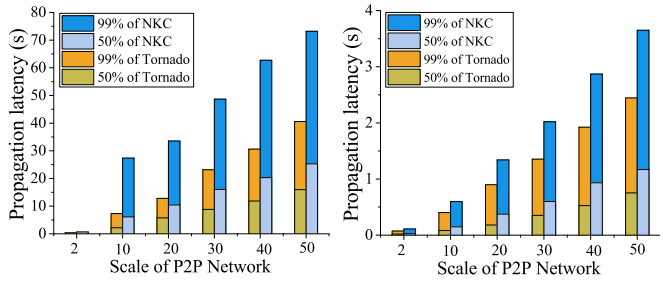


Fig. 5: Required time for mining with varying targets.

Fig. 6: The trend of dynamic weight with epoch.



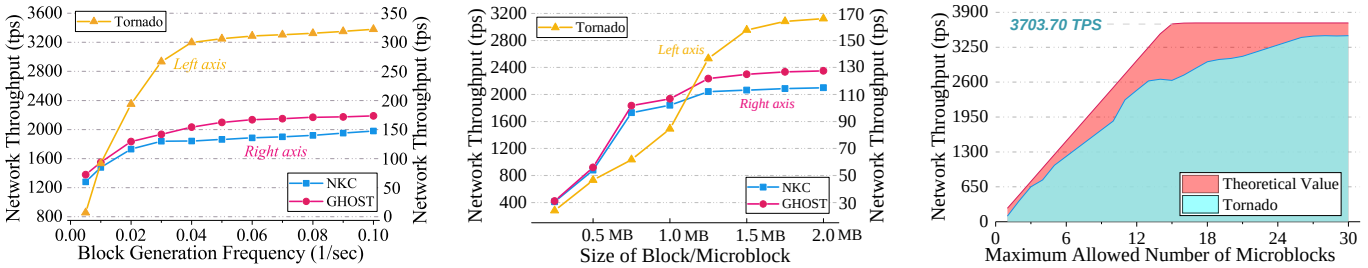
(a) Propagation latency of NKC and Tornado in testbed.

(b) Propagation latency of NKC and Tornado in LAN.

Fig. 7: Inspections of propagation latency.

It should be noted that we choose NKC and GHOST as the baseline approaches because they come from two most well-formed and classic blockchain projects, i.e., Bitcoin and Ethereum. However, these two projects integrate many components which are not the focus of our evaluations, e.g., smart contract engine. In fact, our purpose is to prove the superb performance and applicability of Tornado in heterogeneous IoT. Thus, we extract the core modules from Bitcoin and Ethereum then form NKC and GHOST. The consensus mechanism of both NKC and GHOST is standard-PoW. Viewed from ledger architecture, NKC extends blockchain following a linear manner, in which the longest fork becomes the main-chain. As aforementioned, GHOST weighs every vertex via Eq. (2) and selects the heaviest fork as the main-chain. Eventually, side-chain will be excluded from transaction processing no matter in NKC or GHOST.

2) Simulator: To simulate different scale of workloads in the P2P network of Tornado, we compile the automatic transaction generator. As an abstract wallet software, transaction generator accesses the address list of miners



(a) Comparison of throughput with varying block creation rates.

(b) Comparison of throughput with varying block size limitations.

(c) Comparison of the theoretical peak throughput and actual results.

Fig. 8: Inspections of network throughput and scalability.

and initiates simulated transactions at an adjustable rate. In our testbed, each miner is connected by one transaction generator. Suppose that the efficiency of trans-generator i is τ_i, the network workload can be simply calculated as $\sum \tau_i$ ($\forall i \in$ {transaction generator}).

3) Testbed: We construct a geographically distributed P2P network composed of 50 heterogeneous devices. As shown in TABLE 2, miners that perform different workflows are equipped with wimpy (Raspberry Pi) or brawny (cloud server) devices. The item "hashrate" is defined as the number of hash operations that miner can execute per second, whose unit is mega hashes per second. Moreover, since the network also exhibits heterogeneity, we allocate individual bandwidth for each device. To illustrate the stability of Tornado under different network infrastructures, we also build a 1.5Gbps Local Area Network (LAN).

6.1 Inspections of Mining Difficulty

When describing Co-PoW, we mentioned that microblock miners could solve Eq. (1) within 1s. In fact, the mining difficulty is determined by *target*, i.e., the lower the *target* is, the more hardly miners mine one block. In this part, we evaluate the time demanded by our miners to finish PoW under different *targets*. Same as hashrate measurements in TABLE 2, the PoW modules of NKC, GHOST, and Tornado are all implemented based on Python's hashlib.sha256() library. Nonetheless, the difference is Co-PoW used by Tornado adopts differentiated mining difficulty. Fig. 5 proves that the goal of differentiated mining difficulty can be easily realized by assigning appropriate targets to macroblock and microblock miners, e.g., 229 for the former and 240 for the latter. In such case, heavy workload of macroblock miners defends attacks, while microblock miners create microblocks at high speed for carrying transactions. Further, for the entire P2P network, the creation rate of block (in NKC and GHOST) or macroblock (in Tornado) is also regulated via *target*. This rate will be used below.

6.2 Inspections of Dynamic Weight

Before illustrating the performance of Tornado, we first validate the effectiveness of DW in reflecting data trustworthiness. To stimulate a selfish mining environment, we partition the testbed into honest and malicious parts, whose compositions are shown in Fig. 6. Malicious attackers create secret side-chain following the selfish mining strategy. At *epoch* 1, honest nodes and attackers create own macroblocks B_h and B_m pointing to the same parent. Then, each partition extends one side-chain atop corresponding macroblocks. We compare the DW of B_h and B_m against *epoch*, wherein both Δ_m and Δ_{mi} are set as 1. As shown in Fig. 6, although B_h and B_m share the same DW in *epoch* 1, B_h gradually surpasses B_m since more miners and clients append macroblocks and transactions to its subtree. If attackers broadcast the secret side-chain in *epoch* 6, it will be put behind B_h since the lower DW. Therefore, DW can effectively evaluate the data trustworthiness.

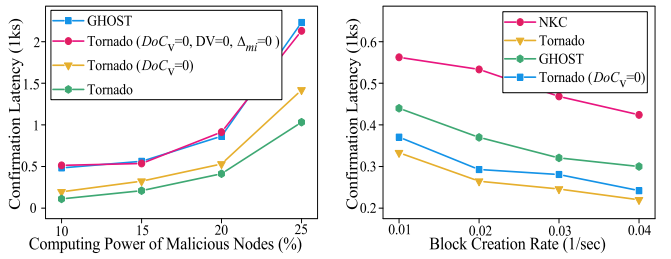
6.3 Inspections of Performance

1) Propagation latency: To evaluate the validity of asynchronized broadcast strategy, we compare the propagation latency of Tornado and NKC (the same as GHOST) under different network conditions. We mainly focus on two states, i.e., the pending macroblock header (for Tornado) or block (for NKC) is synchronized to 1) 50% and 2) 99% of all 50 peers in the P2P network. First, we measure the size of data structures that will be broadcast. Provided that 1000 transactions are carried, traditional blocks in NKC occupy approximately 435.82 KB. As to Tornado, we merely consider macroblock headers because they carry all control information and guide peers to perform workflows. In general, macroblock headers only occupy 208.48B.

As shown in Fig. 7, the propagation latency of Tornado outperforms NKC by 68.14% and 33.02% in the heterogeneous testbed and LAN, respectively. The results clearly display the superiority of broadcasting macroblock headers, especially for the scenarios with limited bandwidth [Fig. 7(a)]. Thanks to the space-structured ledger, Tornado ensures a rapid synchronization of blockchain extensions. Meanwhile, massive transactions are tolerated be received after a certain range of delay, without affecting the local ledger and network security. Furthermore, propagations of Tornado display weak sensitivity when network scaling out, which greatly improves the system scalability.

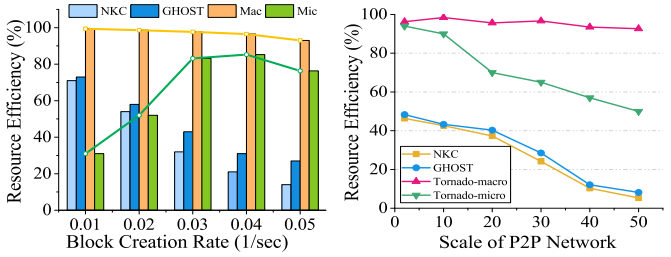
2) Network throughput: To measure the transaction processing speed under Co-PoW, we compare Tornado with NKC and GHOST in terms of network throughput. To guarantee the coverage of results, we conduct extensive experiments by adjusting the following parameters:

- Increasing n (the maximum capacity of microblock or block) from 250KB to 2MB, stepping 250KB.
- Accelerating f (the creation rate of macroblock or block) from 0.01s^{-1} to 0.1s^{-1}, once stepping 0.01s^{-1}.
- Relaxing s (the maximum allowed microblock number in one macroblock) from 1 to 30, stepping 1.



(a) Confirmation latency with varying attaker's power. (b) Confirmation latency with varying block creation rates.

Fig. 9: Inspections of transaction confirmation latency.



(a) Resource efficiency with varying block creation rates. (b) Resource efficiency with varying network scales.

Fig. 10: Inspections of resource efficiency.

The settings of experimental parameters are rational and fit the practical applications. For example, the f of Bitcoin and Ethereum are $0.0016s^{-1}$ and $0.067s^{-1}$, respectively. As to n, Bitcoin and Ethereum are designed as 1MB and 0.02MB. Recently, Bitcoin community is discussing about an increasing n, e.g., 2MB. Following the above settings, we comprehensively evaluates the throughput of Tornado in almost all available environments.

Through the differentiated mining difficulty and parallel workflows, Co-PoW greatly exceeds PoW. As shown in Fig. 8(a) and (b), Tornado reaches over 17.02× that of NKC and 12.98× that of GHOST. With the increment of f, Tornado's throughput experiences rapid improvement, and finally stays at around 3400TPS [Fig. 8(a)]. Recall that the relationship between microblock miners is collaborative, the more microblock miners are, the more concurrent transactions can be packed in parallel. Hence, if we add more microblock miners, this value will undoubtedly keep rising. In Fig. 8(c), when n = 2MB, f = $0.06s^{-1}$, and s = 27, Tornado reaches the maximum throughput of 3464.76TPS.

3) Scalability: As to scalability, we note that the concurrent data broadcast in P2P network exceeds its capability when 1) f = $0.06s^{-1}$ in Fig. 8(a) and 2) n = 1.5MB in Fig. 8(b). This helps explain why the throughput of NKC and GHOST begins to flatten out after these two points. Since NKC and GHOST only confirm the transactions on the main-chain, frequent forks become the key bottleneck hindering their throughput when f is high. Meanwhile, the increasing n also promotes ledger forking since the network causes higher latency to propagate bigger blocks. Limited by flaws on ledger architecture, NKC and GHOST cannot further improve throughput with more peers joining in, which means poor scalability. Conversely, Tornado is unaffected by forks due to the DAG foundation. We measure that leader generally consumes 0.27ms for confirming one pending transaction, which means the peak throughput of Tornado is 3703TPS. As depicted in Fig. 8(c), the theoretical value and Tornado's actual performance converge together. Then, we opine that Tornado is mainly bounded by the leader's capability, rather than ledger architecture or consensus mechanism. More important, the throughput of Tornado will increase linearly as more microblock miners join in the network, showing an outstanding scalability.

4) Transaction Confirmation Latency: The definition of confirmation latency is the time interval that blockchain consumes to confirm one pending transaction. This indicator is tightly related to the QoS of blockchain applications. Similar to NKC and GHOST, confirmations in Tornado are based on probability. In detail, one transaction can be confirmed only if the probability that attackers successfully tamper this transaction is less than the clients' expectation. Suppose that the confidence coefficient is 0.99, which means the probability of successful tampering is less than 0.01. Fig. 9(a) and (b) shows the confirmation latency of Tornado.

We observe that S^{2}GHOST significantly accelerates the transaction confirmations. For the cases where DoC_V is set to 0 [Fig. 9(a) and (b)], the DWA performs weighing only through DV and Δ_{mi}. Therefore, macroblocks carrying more microblocks have a higher chance to be confirmed first, while the impact of macroblocks on confirmation is not used. Thus, the latency optimization is relatively less. If DoC_V, DV, and Δ_{mi} are set to 0, S^{2}GHOST roughly degenerates to GHOST. Confirmation latency is thoroughly optimized when full DW is enabled, which demonstrates the effectiveness of our proposal. Fig. 9(b) illustrates that the value of f will also have an impact on the performance of confirmation latency. Note that in Fig. 9(a), the parameter of x-axis represents the ratio of the total computing power owned by Byzantine macroblock miners to that owned by all macroblock miners.

6.4 Inspections of Resource Efficiency

For individual miners, we define the resource efficiency as: Ω_S/Ω_T, where Ω_T indicates the total number of hash operations executed by miners, Ω_S refers to one part of Ω_T that contributes to successful block (in NKC and GHOST) or macroblock/microblock (in Tornado) creations. Fig. 10(a) illustrates that macroblock miners of Tornado can maintain a resource efficiency higher than 93.42%, regardless of f. Attributed to DAG foundation, the performance increases over 30.56% that of NKC and 27.39% that of GHOST. Limited by s, microblock miners achieve the maximum efficiency when f = 0.04 s^{-1}. The corresponding improvements are 3.04 × that of NKC and 1.74 × that of GHOST, respectively. Owing to high resource efficiency, wimpy devices will be willing to join Tornado, and thus delaying the intensification of Matthew effect [10, 40]. As f increases, the number of conflicting forks lifts up dramatically. In this case, the advantages of our proposal are further enhanced. Moreover, Fig. 10(b) illustrates that the resource efficiency of Tornado declines slower than the scaling of the P2P network, thereby ensuring the high-level stability and scalability.

7 Future Work

In the above sections, we demonstrate the system model, data structures, and novel algorithms of Tornado. Also, we

analyze the security and performance of our proposals in heterogenous IoT scenarios. In this section, some future work which could refine Tornado is discussed. The research points we mention here are also hot topics or open issues for IoT-oriented blockchain.

1) Incentive mechanism: As mentioned in Section 4.2, the introductions of incentive mechanism better measure the trustworthiness of IoT devices. In detail, system could reward devices by tokens or credit if they obey the workflow. Accordingly, the credit will be confiscated as the cost of malicious behaviors, such as lazy leader. Moreover, incentive mechanisms can also combine with Co-PoW to implement a dynamic mining difficulty, which has attracted widespread research interests [6, 10]. With appropriate incentives, IoT devices, especially power-sufficient macroblock miners, will be more willing to invest computing power in maintaining ledger and collaborating with microblock miners.

2) Adaptability: In the early stages of blockchain, the applications are limited to cryptocurrencies, which merely support simple scripts, such as P2PKH. With the Ethereum Virtual Machine (EVM), Ethereum first provisions Turing-complete codes to be executed on blockchain, called smart contracts. In this way, blockchain can significantly enhance its adaptability towards various scenarios and become a distributed operating system. Unfortunately, most well-known high-performance blockchain projects followed a simple model without supporting smart contracts, e.g., Bitcoin-NG [41] and Monoxide [42]. So, modifying Tornado by enabling smart contracts is another meaningful work.

3) Transparency and privacy: Owing to the distributed manners, data in blockchain systems is transparent to all participants. Even outside clients can easily access the blockchain through connected peers. However, IoT scenarios require sensitive data hold confidentiality and is only accessible by authorized devices. To tackle this trade-off, some work proposed the access control schemes to protect vital data while minimizing the impact on decentralization of blockchain [43]. We believe that such mechanisms will further improve the functionality of Tornado.

8 Conclusion

In this paper, we present Tornado, a novel space-structured blockchain system suitable for IoT scenarios. Tornado can effectively overcome the heterogeneity of IoT and accommodate both wimpy and brawny devices. Specifically, we develop the Co-PoW to improve the network throughput. Moreover, the space-structured chain architecture is presented for scalability enhancement. To exploit the resource efficiency of IoT devices, we design an advanced ordering protocol named S^2GHOST with DWA mechanism. The detailed analysis proves the resistance of Tornado facing various attacks. Finally, realistic deployment of 50-node heterogeneous P2P network demonstrates that Tornado comprehensively outperforms NKC and GHOST, thereby enabling blockchain in IoT scenarios.

Acknowledgments

This work is supported by National Key Research and Development Program of China (2018YFB1004700); NSFC (61872195, 61832005, 61572262).

References

[1] M. Khan, X. Wu, X. Xu, and W. Dou, "Big data challenges and opportunities in the hype of industry 4.0," *in Proc. ICC*, Paris, pp. 1-6, 2017.

[2] J. An, and W. Chung, "A novel indoor healthcare with time hopping-based visible light communication," *in Proc. WF-IoT*, Reston, VA, pp. 19-23, 2016.

[3] M. Singh, A. Singh, and S. Kim, "Blockchain: A game changer for securing IoT data," *in Proc. WF-IoT*, Singapore, pp. 51-55, 2018.

[4] M. Ramadan, "Industry 4.0: Development of smart sunroof ambient light manufacturing system for automotive industry," *in Proc. ASET*, Dubai, United Arab Emirates, pp. 1-5, 2019.

[5] K. Wang, X. Qi, L. Shu, D. J. Deng, and J. J. P. C. Rodrigues, "Towards trustworthy crowdsourcing in social internet of things," IEEE Wireless Communications, vol. 23, no. 5, pp. 30-36, October 2016.

[6] J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, "Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3680-3689, June 2019.

[7] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran, "Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5791-5802, June 2019.

[8] H. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," IEEE Internet of Things Journal. doi: 10.1109/JIOT.2019.2920987

[9] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: research issues and challenges," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2188-2204, April 2019.

[10] Y. Liu, K. Wang, Y. Lin, and W. Xu, "Lightchain: A lightweight blockchain system for industrial internt of things," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3571-3581, June 2019.

[11] S. Sankaran, S. Sanju, and K. Achuthan, "Towards realistic energy profiling of blockchains for securing internet of things," *in Proc. ICDCS*, Vienna, pp. 1454-1459, 2018.

[12] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A comprehensive survey of blockchain: from theory to IoT applications and beyond," IEEE Internet of Things Journal, June 2019. doi: 10.1109/JIOT.2019.2922538

[13] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," *in Proc. SIGSAC*, Vienna, Austria, 2016.

[14] W. Chen, Z. Zhang, Z. Hong, J. Wu, S. Maharjan, Z. Zheng, and Y. Zhang, "Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things," IEEE Internet of Things Journal. doi: 10.1109/JIOT.2019.2918296

[15] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," IEEE Transactions on Vehicular Technology, vol. 67, no. 11, pp. 11049-11061, November 2018.

[16] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: A lightweight scalable blockchain for IoT security and privacy," [Online]. Available: https://arxiv.org/pdf/1712.02969.pdf

[17] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in IoT," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4650-4659, June 2019.

[18] I. Farris, L. Militano, M. Nitti, L. Atzori, and A. Iera, "Federated edge-assisted mobile clouds for service provisioning in heterogeneous IoT environments," *in Proc. WF-IoT*, Milan, pp. 591-596, 2015.

[19] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184-1195, April 2018.

[20] D. Sundman, M. Lopez, and L. Wilhelmsson, "Partial on-off keying - a simple means to further improve IoT performance," *in Proc. GIOTS*, Bilbao, Spain, pp. 1-5, 2018.

[21] H. Kaur and A. S. Kushwaha, "A review on integration of big data and IoT," *in Proc. ICCS*, Jalandhar, pp. 200-203, 2018.

[22] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain based cloud data centers," IEEE Cloud Computing, vol. 4, no. 6, pp. 50-59, November 2017.

[23] H. Pirayesh, P. K. Sangdeh, and H. Zeng, "EE-IoT: An energy-efficient IoT communication scheme for WLANs," *in Proc. INFOCOM*, Paris, France, pp. 361-369, 2019.

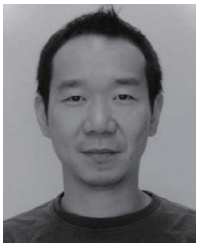
[24] K. Kaushik, and S. Dahiya, "Security and privacy in IoT based ebusiness and retail," *in Proc. SMART*, CMoaradabad, India, pp. 78-81, 2018.
[25] A. C. Panchal, V. M. Khadse, and P. N. Mahalle, "Security issues in IIoT: A comprehensive survey of attacks on IIoT and its countermeasures," *in Proc. GCWCN*, Lonavala, India, pp. 124-130, 2018.
[26] X. Wang, C. Yang, and S. Mao, "PhaseBeat: Exploiting CSI phase data for vital sign monitoring with commodity WiFi devices," *in Proc. ICDCS*, Atlanta, GA, pp. 1230-1239, 2017.
[27] H. Li, K. Wang, T. Miyazaki, C. Xu, S. Guo, and Y. Sun, "Trust-enhanced content delivery in blockchain-based information-centric networking," IEEE Network, 2019. doi: 10.1109/MNET.2019.1800299
[28] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3527-3537, June 2019.
[29] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (IIoT) systems: a deep reinforcement learning approach," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3559-3570, June 2019.
[30] Pay to public key hash (P2PKH) script, [Online]. Available: https://bitcoin.org/en/transactions-guide
[31] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," *in Proc. FCDS*, Springer Berlin Heidelberg, 2015. doi: 10.1007/978-3-662-47854-7_33
[32] K. Karlsson, "Vegvisir: A partition-tolerant blockchain for the internet-of-things," *in Prof. ICDCS*, Vienna, pp. 1150-1158, 2018.
[33] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," *in Proc. CCS*, CM, New York, pp. 931-948, 2018.
[34] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 4, pp. 870-882, April 2019.
[35] M. Vukoli, "The quest for scalable blockchain fabric: proof-of-work vs. bft replication," *in Proc. iNetSec*, Zurich, Switzerland, pp. 112-125, 2015.
[36] M. Castro and B. Liskov, "Practical byzantine fault tolerance," *in Proc. OSDI*, New Orleans, Louisiana, USA, pp. 173-186, 1999.
[37] N. Satoshi, "Bitcoin: A peer-to-peer electronic cash system," [Online]. Available: https://bitcoin.org/bitcoin.pdf
[38] Y. Sompolnsky and Zohar, "Phantom, GHOSTDAG: Two scalable blockDAG protocols," [Online]. Available: https://eprint.iacr.org/2018/104.pdf
[39] I. Eyal and E. G. Sierer, "Majority is not enough: Bitcoin mining is vulnerable," [Online]. Available: https://arxiv.org/abs/1311.0243
[40] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial internet of things architecture: An energy-efficient perspective," IEEE Communications Magazine, vol. 54, no. 12, pp. 48-54, December 2016.
[41] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," *in Proc. NSDI*, Santa Clara, CA, pp. 45-59, 2016.
[42] J. Wang and H. Wang, "Monoxide: Scale out blockchains with synchronous consensus zones," *in Proc. NSDI*, Boston, MA, pp. 95-112, 2019.
[43] J. Huang, L. Kong, G. Chen, L. Cheng, K. Wu, and X. Liu, "B-IoT: Blockchain driven internet of things with credit-based consensus mechanism," *in Proc. ICDCS*, Dallas, Texas, pp. 1348-1357, 2019.

Yinqiu Liu is working toward the undergraduate degree in the College of Internet of Things, Nanjing University of Posts and Telecommunications, China. His current research interests include wireless communications, Internet of Things, and blockchain.

Kun Wang (M'13-SM'17) received two Ph.D. degrees from Nanjing University of Posts and Telecommunications, China in 2009 and from the University of Aizu, Japan in 2018, respectively, both in Computer Science. He was a Postdoc Fellow in UCLA, USA from 2013 to 2015, and a Research Fellow in the Hong Kong Polytechnic University, Hong Kong, from 2017 to 2018. He is currently a Research Fellow in UCLA. His current research interests are mainly in the area of big data, wireless communications and networking, energy Internet, and information security technologies. He is the recipient of Best Paper Award at IEEE GLOBECOM16. He serves as Associate Editor of IEEE Access, Editor of Journal of Network and Computer Applications, and Guest Editors of IEEE Network, IEEE Access, Future Generation Computer Systems, Peer-to-Peer Networking and Applications, and Journal of Internet Technology.

Kai Qian is a postgraduate student in Information Network at Nanjing University of Posts and Telecommunications, China. His current research interests include network security, blockchain systems, big data, and Internet of Things.

Miao Du is a postgraduate student in Information Network at Nanjing University of Posts and Telecommunications, China. His current research interests include network security, game theory, smart grid communications, and blockchain systems.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa and was a professor with the University of Aizu. He is a full professor with the Department of Computing, The Hong Kong Polytechnic University. His research interests include big data, cloud computing and networking, and distributed systems with more than 400 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. He was an associate editor of the IEEE Transactions on Parallel and Distributed Systems and an IEEE ComSoc distinguished lecturer. He is now on the editorial board of the IEEE Transactions on Emerging Topics in Computing, the IEEE Transactions on Sustainable Computing, the IEEE Transactions on Green Communications and Networking, and the IEEE Communications. He also served as general, TPC and symposium chair for numerous IEEE conferences. He currently serves as an officer for several IEEE ComSoc Technical Committees and a director in the ComSoc Board of Governors. He is a senior member of the IEEE.