PoBT: A Lightweight Consensus Algorithm for Scalable IoT Business Blockchain

Sujit Biswas, *Member, IEEE*, Kashif Sharif, *Member, IEEE*, Fan Li, *Member, IEEE*, Sabita Maharjan, *Senior Member, IEEE*, Saraju P. Mohanty, *Senior Member, IEEE*, and Yu Wang, *Fellow, IEEE*

***Abstract*—Efficient and smart business processes are heavily dependent on the Internet of Things (IoT) networks, where end-to-end optimization is critical to the success of the whole ecosystem. These systems, including industrial, healthcare, and others, are large scale complex networks of heterogeneous devices. This introduces many security and access control challenges. Blockchain has emerged as an effective solution for addressing several such challenges. However, the basic algorithms used in the business blockchain are not feasible for large scale IoT systems. To make them scalable for IoT, the complex consensus-based security has to be downgraded. In this article, we propose a novel lightweight proof of block and trade (PoBT) consensus algorithm for IoT blockchain and its integration framework. This solution allows the validation of trades as well as blocks with reduced computation time. Also, we present a ledger distribution mechanism to decrease the memory requirements of IoT nodes. The analysis and evaluation of security aspects, computation time, memory, and bandwidth requirements show significant improvement in the performance of the overall system.**

***Index Terms*—Blockchain, consensus, distributed ledger technology, Internet of Things (IoT), interoperability, ledger size, scalability, transaction rate.**

I. Introduction

SMART systems for industrial automation, e-health, logistics, etc., aim at providing efficient solutions for business processes by leveraging the benefits of the Internet of Things (IoT). Modern industries comprise smart production systems, global value chain networks (supply chain, services, marketing, etc.), and end-to-end value chain support that includes privacy and transaction security [1]. These and other similar smart services are implemented through large scale complex Industrial IoT (IIoT) systems, to automate and optimize for the better quality-of-service and resource utilization [2]. However, these advantages have a cost associated with them. The complex, interconnected, and heterogeneous networks are vulnerable to cyberattacks. Handling intelligent unstructured data generating equipment, higher standards of data acquisition, integrating heterogeneous data in a unified system by generic protocols, and access control of industrial networks are some of the critical challenges in smart systems [3]. Moreover, to ensure ubiquitous communication, knowledge-based intelligent manufacturing, integration of heterogeneous data resources, and device interoperability have added a new dimension to these challenges [4].

Blockchain (BC) is a distributed ledger technology that provides a secure way of making and recording transactions and contracts. It is consensus driven and trustless but offers highly secure, immutable, and encrypted record keeping mechanism. It has evolved from Bitcoin cryptocurrency [5], which is a public, trustless, and anonymous but a highly secure chain. This has further led to the introduction of private blockchain or business BC (BBC) [6], which can be utilized in industrial operations. BBC is a promising approach with immense potential to enhance and optimize different parts of smart processes where information or data are exchanged.

IoT and more specifically IIoT can significantly benefit from distributed ledger technologies for data exchange, access control, and management. Consider an assembly line IoT device network, which has to report data for quality control. Some of these devices can only be data originators, while other devices may be able to process and take corrective measures in response to them. This is a prime application of blockchain to exchange information among devices in a secure and accountable manner. This exchange is quite similar to cryptocurrencies, however, rather than token, the devices exchange data (or digital assets). This exchange may have to strictly controlled among specific devices, data may be immutable, and the overall process auditable. The traditional centralized systems (even if they are implemented in the cloud) do not provide such facilities. Relational database systems by design are not made for such applications. Hence, the blockchain technology can be effectively utilized in IoT [7]. However, the consensus formation algorithms used in the

Manuscript received March 18, 2019; revised September 5, 2019, October 8, 2019, and November 5, 2019; accepted December 3, 2019. Date of publication December 6, 2019; date of current version March 12, 2020. The work of F. Li was supported in part by the National Natural Science Foundation of China under Grant 61772077 and Grant 61432015, and in part by the Beijing Natural Science Foundation under Grant 4192051. *(Corresponding authors: Kashif Sharif; Fan Li.)*

S. Biswas, K. Sharif, and F. Li are with the School of Computer Science and the Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, Beijing Institute of Technology, Beijing 100086, China (e-mail: sujitedu@bit.edu.cn; kashif@bit.edu.cn; fli@bit.edu.cn).

S. Maharjan is with the Center for Resilient Networks and Applications, Simula Metropolitan Center for Digital Engineering, 0167 Oslo, Norway, and also with the Department of Informatics, University of Oslo, 0316 Oslo, Norway (e-mail: sabita@simula.no).

S. P. Mohanty is with the Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207 USA (e-mail: saraju.mohanty@unt.edu).

Y. Wang is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: wangyu@temple.edu).

Digital Object Identifier 10.1109/JIOT.2019.2958077

traditional blockchains cannot be applied here, as they are extensively resource consuming, while IoT devices are the resource constrained. Moreover, scalability and efficiency are major performance metrics for any IoT system, while BC consensus creates a bottleneck for them [8], [9].

Our motivation in this article is to develop a novel lightweight consensus algorithm targeted at BBCs for IoT solutions. The proposed solution does not have the mathematical complexity of a mining algorithm, and at the same time, it does not compromise on the security and verification of trades. Moreover, the algorithm is scalable for large IoT systems and can easily be integrated with different BBC solutions. In light of this, our contributions in this article are multifold.

1) We propose proof of block and trade (PoBT), a set of novel algorithms specifically designed for use with IoT blockchain, which not only validates the trades but additionally validates the blocks before they are committed to the ledger.
2) We present a complete working solution for the integration of the proposed consensus algorithm with the Hyperledger Fabric framework.
3) We present a novel local trade process for scalability and devise solutions for anomalous timeout behavior of nodes.
4) We have implemented the solution and conducted extensive experiments for the evaluation of computation time, memory, and bandwidth requirements to show its efficiency.

The remainder of this article is organized into six sections. Section II presents the background and existing consensus algorithms in BBC and their limitations. This is followed by the system design of our proposed blockchain for the IoT system in Section III. Section IV presents the working of PoBT, while scalability and timeout anomalies are discussed in Section V. The analysis and evaluation of the system are provided in Section VI. Section VII concludes this article.

II. Background and Related Works

In this section, we first present background information on blockchain types, their application and challenges in IoT, and then discuss the related works.

A. Blockchain Types

Blockchains can be divided into different categories based on two main aspects, i.e., *Application* and *Openness* [6], [10]–[12]. In the first classification based on its application, blockchains can be either for cryptocurrencies or for business processes, such as e-voting, asset tracking, assembly line monitoring, etc. It is important to note that the latter has no currency involved, rather the devices exchange information (as a trade). The second classification can be done based on the openness of the system; i.e., public, consortium/federated, or private blockchains. Public blockchains do not have any access control or restriction on users or peers. Anyone can join the network, initiate trades, or become a peer. They are more suitable for cryptocurrencies. Contrary to this, private (or permissioned) blockchains implement strict access control for nodes joining the network. In a closed network, such as an assembly line, health monitoring, or logistical tracking, each user must be registered and authorized, and hence private blockchain is a more suitable solution. Consortium blockchains are broader than private where a group of organizations is part of the chain. Business BC is usually private, while crypto chains can be private or public in nature. As the business applications do not have currency involved, hence the concept of a miner is modified to that of a peer. A peer does not need to be monetarily incentivized to work, rather it works under the organization.

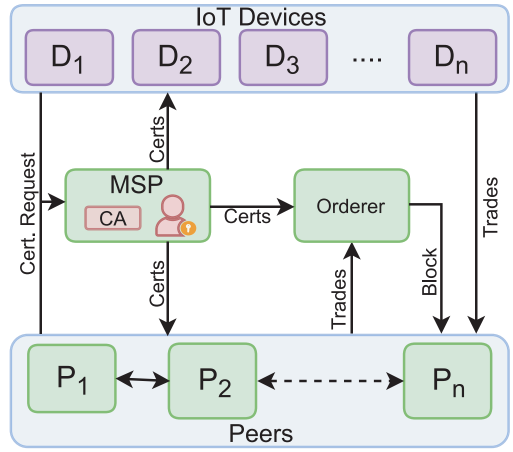


Fig. 1. Generic IoT BBC process.

B. Blockchain for IoT

The BBC solutions adopted for IoT platforms are considerably different from the crypto/public blockchains [13], [14]. The core concepts are the same, but the component integration and algorithms vary to a large extent, such as consensus formation, ordering, etc., IoT and industrial IoT applications utilize these solutions in several processes. A generic IoT BBC process is depicted in Fig. 1. IoT nodes generate trades (transactions) that can contain data or information, that can be shared by other IoT nodes within or outside the local network. Each node is linked with a membership service provider (MSP), which is comprised of an administrator and certificate authority (CA) responsible for providing keys, signature, certificates ($\mathtt{CA}_{\mathtt{cert}}$), and configuration information. Peers are specialized IoT nodes, which have enough resources to execute consensus algorithms and maintain the distributed ledger. Orderer is another kind of node, which is responsible for grouping all endorsed/approved trades into a newly generated block. Chaincode is deployed on the peer nodes for verification of transaction agreements between different IoT devices. IoT nodes generate trades in the context of previously deployed chaincode/smart contracts (SCs) through specific channels (a private subnet of communication between applications of two or more members). Every chaincode-verified successful trade is stored into the ledger as an element of a block, which is done through the ordering services of an orderer. The orderer waits for a specific amount of time (batch time or block time) for new valid trades. At the batch

TABLE I
COMPARISON OF CONSENSUS ALGORITHMS FOR BBC SOLUTIONS

Consensus Algorithm(s)	Platform	Purpose	Description
Ordering Service (Kafka)	Hyperledger Fabric [13]	General	Atomic broadcast service for consumption of nodes
Trusted Validator (Round Rabin)	Multichain [15], Parity [16]	General	Used as validator per block, instead of multiple validators
Raft	Quorum [17], Corda [18]	General, Digital asset	First elects a leader node which is responsible for decisions
Trusted Validators (majority)	Hydra Chain [14], BigchainDB [19]	General, Digital asset	Relies on a set of validators, where $\frac{1}{3}$ must be byzantine
Single Validator	OpenChain [20]	Digital asset	Trades accepted & validated by a node, observers can read
Tangle Consensus	IOTA [21]	Digital asset	Coordination & distribution based industry payment system
Proof of Elapsed Time (PoET)	Sawtooth Lake [22]	General	Prevents high resource utilization and energy consumption
RBFT	Hyperledger Indy [23]	General	Primary replica of multiple trades executed on diff. nodes
Sumeragi	Hyperledger Iroha [23]	General	Validator is reputation based; performed on individual trades

timeout, the orderer closes the block and forwards the new block to all connected peers. All peers verify their credentials and update their respective ledger. A fundamental assumption exists in this framework, that the CA and Orderer are trusted and secure.

C. Blockchain Consensus and Challenges

Public blockchains use proof of work (PoW) [5] or a similar algorithm to solve complex mathematical problems (mining) to create a new block, which is both resource and time consuming [24]. In the mining process, which is a key element of security, public third parties (miners) mine blocks for coin incentives, which may not be part of a business process using IoT devices for data exchange. IoT systems need highly efficient consensus algorithms without compromising security features. Transaction per second (TPS) and the rapid growth of ledger in line with the trades generated by IoT devices are different from the public blockchain. Although private blockchain(s) have evolved to address some of these limitations, complete and optimized solutions are not yet available.

D. Consensus Algorithms in Existing Business Blockchains

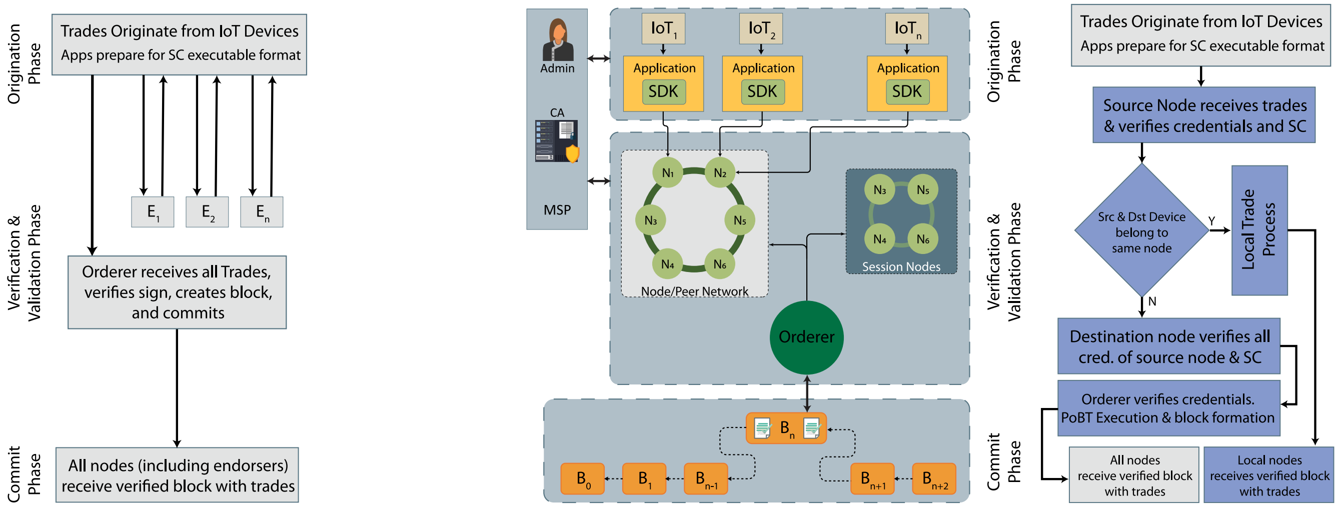
There are only a handful of BBC solutions available and are mainly developed by industry. Many of these solutions maintain the generic characteristics of a blockchain, i.e., a distributed ledger, some form of consensus algorithm, and a P2P network structure. The differences in implementation, algorithms, and processes are the elements that create security and scalability benchmarks for them. In order to maintain the basic features, these solutions have to enforce trade approval and new block creation processes. Table I lists some of the consensus algorithms and associated platforms.

It is important to note that the objective of this article is solely focused on the business solutions for IoT systems and not cryptocurrency chains. Hyperledger [13] is a leading platform for IoT-based business solutions and processes using blockchain. It implements five frameworks intended for different types of environments and consensus mechanisms. These are Fabric, Sawtooth, Burrow, Iroha, and Indy. Here, we focus on two major implementations, Fabric [23] and Sawtooth [22], which can be used in the IoT domain.

Fabric uses the Apache Kafka protocol [25] for ordering the trades into a block. The number of endorsing nodes N_e is fixed from the total number of nodes N. Based on the customization policy, the nodes N_v are actually selected for verification of a given trade. Hence, $N_v \subseteq N_e \subseteq N$, where N_v is typically a very small number of nodes as compared to N. All incoming trades from endorsers are grouped into a block by the orderer using the Kafka ordering services. In essence, Kafka is an ordering service that aims to provide a unified, high throughput, low-latency platform for handling real-time data feeds. Fabric 1.0 does not employ any byzantine fault tolerant (BFT) ordering service and supports only crash faults based on Kafka. Invalid trades may also get added to the ledger if a rouge node in the network sends it to the orderer [26]. In conclusion, it does not consider security measures at block creation, and the number of endorsers is fixed (through chain-code) which can be as small as two. Similar to Fabric, Parity also involves a fixed number of nodes for trade verification and block creation.

Proof of elapsed time (PoET) is used in Sawtooth [22], where network waits a random amount of time for the creation of a new block, and the first participant that finishes waiting is elected as a leader among the member nodes for that specific block. The cost of regulating the election process must be proportional to its potential returns. Therefore, devices that desire to contribute to this are required to invest a substantial amount of resources. Furthermore, the legitimacy of every election must be validated by each member of the community. It works like PoW, except the fact that the whole population N participates in a mathematical puzzle-based leader selection and requires nodes to invest resources, which makes it costly and time consuming.

Consensus algorithms proposed in literature other than Hyperledger are mainly focused toward public crypto blockchains, and hence cannot be implemented for private BBCs, especially for IoT. Wang *et al.* [27] proposed a credit-delegated BFT (CDBFT) scheme for voting rewards, punishments, and credit evaluations. However, in BBC, the participating nodes are authenticated and trusted, thus reward and punishment mechanisms are of less importance than



(a) (b)

Fig. 2. (a) Hyperledger Fabric's verification process. (b) Framework and processes of the proposed system.

scalability and ledger expansion problems. Similarly, proof of authentication [28], [29] discusses block validation for private or permissioned blockchains based on trust values. However, if a node is compromised it may sacrifice its trust value and validate a malicious trade, thus injecting it into the ledger. Kang *et al.* [30] proposed a two-stage soft security enhancement solution for block verification and miner selection in the Internet of Vehicles blockchain. The objective is to prevent internal collusion among active miners and standby miners in a public blockchain. An *et al.* [31] presented a consensus mechanism for the quality control of crowd sensed data. Although the work addresses scalability and security aspects, it is not directly applicable in private IoT BBCs.

Based on the limitations of Hyperledger Fabric and other state-of-the-art solutions in literature, an ideal solution should validate trades as well as the blocks, to provide the maximum level of security, while optimizing the number of endorsers to keep the overhead to the minimum level, in a private BBC. In the proposed PoBT scheme, every new block is validated through a variable number of nodes by solving a simpler mathematical puzzle which is less computation-intensive but ensures the security of the same level. Trades are cross verified by involved trade nodes. Hence, it is a twofold checking mechanism, which, without compromising on the security, ensures a higher number of TPS. Furthermore, it distinguishes between local trades and global trades, hence addresses the ledger scalability issues also.

III. Blockchain for IoT: System Design

The overall proposed system design of blockchain integration in an IoT network is shown in Fig. 2. It is important to note that the blockchain is a relatively new solution with very few real-world implementations. We consider Hyperledger Fabric as the baseline solution for adoption. However, the framework presented here is not limited to it and can be integrated with other solutions also. The complete process of blockchain in IoT consists of three phases: 1) trade origination; 2) verification and validation; and 3) committing phase. Fig. 2(a) shows the generic working process of Hyperledger Fabric, which is quite similar to the working shown in Fig. 1. Fig. 2(b) shows the initial connectivity of IoT devices to the blockchain *nodes*, which act as peers or potential endorsers. A trusted CA as part of a larger MSP also exits in the system along with an administrator. In Fig. 2(b), we depict the proposed workflow (in contrast to Hyperledger Fabric) based on PoBT. Before explaining each phase, we define the following terms used in this article.

Device: In the proposed system, a device is any IoT equipment which is capable of generating or receiving blockchain trades (transactions). For example, a smartwatch, a sensor on an assembly line, a decision making intelligent IoT, etc. As shown in Figs. 1 and 2(b), the IoT devices represent them in our system.

Node: A node (or peer) is part of the blockchain core network. It is a device capable of executing the consensus process and store the ledger. An IoT device is always connected to a node, which processes the trades originating from that device. Nodes are shown in Fig. 1 as the peers (P_1, P_2, P_n), and as nodes (N_1, N_2, $\ldots$) in Fig. 2(b).

User: A user generally refers to a human who is taking part in the system, however, in the proposed architecture, there is no human involvement except the administrator. An IoT may be operated/owned by a human user, but there is no interaction of the user in the blockchain process for generation, validation, or storage of trades. The administrator is part of the design as shown in Fig. 2(b) so that the system can be initialized and MSP can be maintained.

A. Trade Origination Phase

Ubiquitous IoT devices from different vendors are utilized in a production environment in smart environments that generate data in various formats. As an example, temperature sensors or other monitors attached to critical automation equipment may have different measuring units. It is challenging to receive multistructured data and then confirm it to a format executable on chaincode. Besides, these devices are also resource constrained and thus cannot act as blockchain nodes themselves. Hence, they are linked to a node N_i, which acts as their blockchain node. In any given system, N_i can be preconfigured, or the IoT devices can be programmed to locate the nearest one and establish a secure connection to it. We consider that all N_i have enough resources to execute the desired blockchain functions.

1) Trade Proposal Preparation: Applications on IoT devices collect data that is to be exchanged (as a trade). It is then formatted using the software development kit (SDK) for execution on chaincode. This trade proposal includes trade data as payload along with device signature, destination public address, and corresponding certificates. For example, a quality control device on an assembly line may generate trades that contain product statistics as data to be stored in the blockchain. Hence, the application on the device interacts with CA, which generates enrollment certificates (`eCert`) for enrollment into the blockchain network. As shown in Fig. 2(b), *admin* is the sole authority to approve IoT device integration and chaincode installation, while CA is responsible for generating all credentials. These are generated for all entities: admin, devices, nodes, and applications.

2) Trade Proposal Execution: As each device has a connecting blockchain node, a trade proposal is sent to it for execution through the channel. Every application is provided with a channel, which acts as a logical communication tunnel between the application and the node. Authentication and authorization to transact are strictly bound to the channel, hence a device cannot access any other node or execute trades which are prohibited on a given channel. MSP is responsible for initializing and maintaining the channels. It is important to note that many IoT devices are connected to a node, but in a given blocking session, each node is restricted to one trade to avoid double spending [32].

B. Verification and Validation Phase

All incoming trade verifications depend on the proper recognition of devices, users, rights, etc. Validation is the second step after verification where trades are validated, depending on the terms and conditions specified in chaincode or SC. Verification process is executed using certificates, i.e., Transport layer security certificate ($\mathtt{TLS}_{\mathtt{cert}}$) for communication and `eCert` for enrollment.

1) Authenticity of Users and Devices: There are two types of participating elements: 1) nodes and 2) devices. Admin is a trusted authority whose credentials like `eCert`, *sign*, *keys*, $\mathtt{TLS}_{\mathtt{cert}}$, and $\mathtt{CA}_{\mathtt{cert}}$ are generated when the network is instantiated. Admin object provides applications with proper *eCert* to register new nodes/devices. Similarly, it also communicates with the CA, to enroll new nodes, who may query or add blocks to the ledger. This process ensures that no unidentified user or device is allowed to join the network without proper identification and authentication.

2) Smart Contract Deployment: Chaincode is installed on a *node* by admin and then instantiated on a channel with an identity (including name and version) fulfilling its instantiation policy. In Hyperledger, the installation and instantiation of chaincode follow the same trade flow as a normal invocation, i.e., endorse, order, validate, and commit. However, after installation, changes can be made which creates a major security issue, as trade validation is directly dependent on the SC. Chaincode plays an extremely important role in the whole process as it enforces trade policies. These policies primarily enforce trade execution rules between the participating devices. In our proposal, any change to the SC follows the same consensus process as that of individual trades. This ensures that at least 51% of the nodes agree to change the SC.

3) Access Right Verification: Access rights are primarily defined through the channel and then the SC script. Devices are only allowed to communicate with nodes via a channel where both devices and nodes have to use the MSP generated credentials. As the channel is assigned by *admin*, it becomes impossible to access nodes/devices which are not connected.

4) Verification Through Neighbors: In the existing BBC (Fabric), the SC dictates which and how many neighbors verify a trade. This can be as small as two nodes. On the contrary, the main principle of blockchain is to establish consensus using a large set of nodes. In our proposed system, we use a ratio of total nodes as endorsing nodes based on the trade submission in a session. By doing this, we keep the endorsement overhead less than crypto chains but can guarantee better security than the existing business chains.

5) Trade Acceptance or Rejection: In the existing BBCs, trades pass through endorsement (approved by neighbors) and are accepted as valid trades. The orderer receives concurrent trades from many nodes according to its batch timeout and creates a new block that is added to the ledger. As shown in Fig. 2(a), endorsing nodes E_i endorse the trade by verifying its signatures and SC requirements, and the orderer creates the block. At block closing time there is no validation or rejection, which means that if a malicious node sends a forged trade to the orderer, it will be added to the block. Bitcoin and similar systems, implement the consensus algorithm at block creation time, which ensures that any malicious trade is not added to the ledger. However, as discussed earlier, in IoT real-time systems the transaction rate cannot afford long delays in consensus formation.

Our proposed framework is depicted in Fig. 2(b). The IoT devices connected to the same node are not part of global consensus, and are handled by the process defined in Section V. For trades among devices that are connected to different nodes, first, they are endorsed by participating nodes, and then consensus (PoBT) is formed by multiple nodes involved in the specific blocking session. This process is described in detail in Section IV.

C. Commit Phase

This is the final stage of trade processing. When a block is finalized through the consensus PoBT, then it is completely

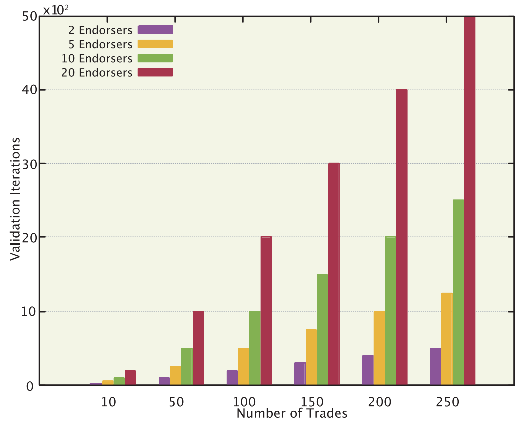


Fig. 3. Effect of endorsers in baseline Hyperledger Fabric.

ready to be distributed to all nodes in the network, which they can add to their ledgers.

Block Distribution: The orderer approves the new block only if the PoBT algorithm returns true, and then distributes it to all connected nodes in the network along with the signature of the orderer. Every node verifies the signature and adds the block to its ledger.

World State: Finally, the added block is synchronized with the ledger and the world state is updated.

IV. Consensus: Proof of Block and Trade

The trades which become part of a block and are inserted into a ledger are immutable. A single malicious or inaccurate trade jeopardizes the complete blockchain integrity. In this section, we present the working of our consensus algorithm for BBC in an IoT environment. First, we elaborate on the technical challenges of such an algorithm, and then we describe the working principle of PoBT in detail.

A. Challenges of Consensus Formation

Every blockchain system needs to allow nodes to present new transactions for validation, and to facilitate the election of candidate blocks. The election is run through a consensus mechanism. For a given consensus algorithm, the number of messages required to reach consensus regarding block election increases significantly with each added endorser. Every single endorser must contact at least more than half of the entire population, and all of those must perform the same validation steps (the validation code needs to be deterministic [33]), hence system transaction throughput decreases and latency increases. Fig. 3 shows the difference in the number of iterative validations performed in a blockchain system to form a consensus in the presence of a varying number of endorsers and trades. Here, more endorsers require more trade verification executions, which will directly reduce scalability and increase latency. For example, when the number of trades is 250, the difference in the number of validations is more than 4000 between 2 and 20 endorsers. A highly scalable system should be able to handle a large trade volume, without compromising the block consensus security and TPS. The limitations of TPS

Algorithm 1: Session Node Selection

Struct *Node* **contains**
  int addr; byte trd, arg;
end
initialize Node $L[] \leftarrow Null$; $i \leftarrow 0$;
while $(!session\_timeout \wedge !active\_blocking)$ **do**
  Initialize $k \leftarrow 0$; $src, dst \leftarrow -1$;
  Receive $Tr^i(M, Sign(N_{src}, N_{dst.}), N_{src}^{PA}, N_{dst}^{PA})$;
  for $k < i$ **do**
    if $N_{src}^{PA} \equiv L[k].addr$ **then**
      $src \leftarrow k$;
    end
    if $N_{dst}^{PA} \equiv L[k].addr$ **then**
      $dst \leftarrow k$;
    end
  end
  if $src \geq 0$ **then**
    $L[src].trd++$;
  else
    $L[i].addr \leftarrow N_{src}^{PA}$; $L[i].trd \leftarrow 1$; $i++$;
  end
  if $dst \geq 0$ **then**
    $L[dst].trd++$;
  else
    $L[i].addr \leftarrow N_{dst}^{PA}$; $L[i].trd \leftarrow 1$; $i++$;
  end
end
return $L[]$;

in Hyperledger BBC have been addressed by removing the consensus and allowing trade endorsing nodes to be as less as two nodes [34]. Although it extensively reduces the overhead (Fig. 3), the number of constant endorsers e may compromise the security. For example, in an $n = 100$ node environment with $e = 2$, only two endorsers are required to validate a trade. Hence, the probability of success for malicious nodes to insert a compromised trade in a chain becomes 98%.

In order to address this issue, instead of using a repeated endorsement process by all nodes or at least 51% of the nodes, this article uses a subset of network participants for endorsement. The size of this subset is dependent on the number of participating nodes in a given block. As the endorsers are dynamically selected, hence the attacker cannot pre-empt which nodes to compromise to validate the illegal trades. Moreover, the endorsers use a lightweight algorithm for validation, thus the computational complexity is significantly reduced. The resulting effect gives a higher level of security, lower overhead from messaging, a less computational requirement of endorsing nodes, and a higher transaction rate for the system.

B. Proof of Block and Trade: Solution for Consensus

In order to solve the challenge discussed above, here we present PoBT for BBC used in the IoT systems. To improve the

Algorithm 2: PoBT Algorithm

Input : $(Tr, L[])$
Output: *True Or False*
1 set $j \leftarrow 0$; $p \leftarrow 0$; $i \leftarrow count(L)$;
2 **while** $j < i$ **do**
3 set $R^j \leftarrow Random.serial(R_n)$; ▷ $1 \le R_n \le i$
4 Group all trades $\acute{Tr}^j \leftarrow L[j].addr$;
5 set $L[j].arg \leftarrow R^j$; $L[j].trd \leftarrow R^j$;
6 $j++$;
7 **end**
8 $Verify(\acute{Tr}^{L[j].addr}, R^j) \rightarrow N_s^j$ //$(\exists N_s^j \in N_s) \wedge N_s^j \neq N_s^{L[j].addr}$ ▷ Internal Condition
9 Initialize $total \leftarrow (1 + \frac{i-1}{2}) \times i$; $k \leftarrow 0$; $n \leftarrow \frac{i}{2} + 1$
10 **while** $(k < n \wedge !timeout)$ **do**
11 set $m \leftarrow 0$;
12 Receive $R^j, N_s^{sign}, N_s^{addr} \leftarrow verify(R^j, N_s^{sign}, N_s^{addr})$
13 **for** $m < j$ **do**
14 **if** $(L[m].addr \equiv N_s^{addr}) \wedge (R^j \equiv L[m].arg)$ **then**
15 $sum \leftarrow sum + L[m].arg$;
16 $k++$; *break*;
17 **end**
18 **end**
19 **end**
20 Set $sum \leftarrow sum + Faulty\ nodes(R^x) + non\ responsive(R^y)$
21 **if** $(sum \equiv total) \wedge (k \ge n)$ **then**
22 *return True*;
23 **else**
24 *return False*;
25 **end**

Algorithm 3: N_s Verification Process

Input : $(\acute{Tr}, R^j)$
Output: $(R^j, N_s^{sign}, N_s^{addr})$
Set $n \leftarrow count(\acute{Tr})$; $k \leftarrow 0$; $temp2 \leftarrow 0$
for $k < n$ **do**
if $Sign(N_{src}^{\acute{Tr}[k]} \wedge N_{dst}^{\acute{Tr}[k]}) \wedge (chaincode^{\acute{Tr}[k]})$ *Not verified* **then**
set $temp2 \leftarrow 1$; *break*;
else
$temp2 \leftarrow 0$;
end
end
if $temp2 \equiv 0$ **then**
return $verify(R^j, N_s^{sign}, N_s^{addr})$;
else
return $verify(R^j \leftarrow -1)$;
end

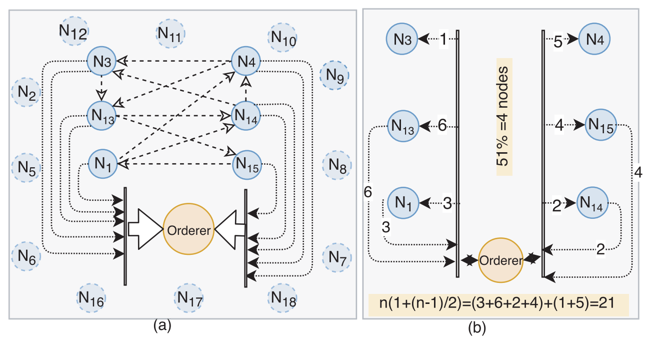


Fig. 4. (a) Session node selection. (b) Consensus process.

scalability of consensus and increase the security of consensus-less Fabric, we utilize a hybrid mechanism. For simplicity, we divide the complete process into two parts as trade verification and then consensus formation. Following this, we compute the processing time.

1) Trade Verification: In the proposed consensus model, source node ($\mathtt{N_{src}}$) receives trades from its associated IoT devices $\mathtt{D_s}$. It then verifies the SC (for permission to trade) and checks if the destination IoT device $\mathtt{D_d}$ is also associated with it. If that is the case, then a Local Consensus Process is executed, which allows ledger scalability as explained in Section V. Otherwise, the trade is forwarded to the destination node ($\mathtt{N_{dst}}$) connected to the destination device. $\mathtt{N_{dst}}$ also verifies the SC (for cross-checking the permission to trade), and if approved forwards the trade to the orderer for consensus formation. Hence, trade verification is limited to the nodes directly involved in the trade. This significantly reduces the information exchange, time required, and control overhead, without compromising the security.

2) Consensus Formation: The orderer performs the consensus on a candidate block which contains several verified trades it collects in a given amount of time. This time is represented as *session_timeout*. During this time, Algorithm 1 is used to build a list of session network members. This list keeps track of individual $\mathtt{N_{dst}}$ or $\mathtt{N_{src}}$, and the number of trades they are participating in. The algorithm uses a specialized data structure to maintain the node's public address, the number of trades it is involved in, and argument data used in the block formation process (lines 1–3). During the session, the algorithm receives complete trade from $\mathtt{N_{dst}}$ and adds it to the list (line 7). Finally, the list L (line 27) reflects all source or destination nodes, and the number of trades they are involved in. It is important to note that the block creation time may vary, hence we enforce that even if *session_timeout* expires, the trade collection continues until the previous block has been committed (*active_blocking*).

After complete execution of Algorithm 1, a set of session nodes $\mathtt{N_s} \subseteq \mathtt{N}$ is available. Fig. 4(a) shows only those nodes which have trades to be reported to the orderer. They form the session network and will participate in the consensus formation. List L is then forwarded to Algorithm 2 along with $\mathtt{Tr}$ which is a complete set of trades in this session. In Algorithm 2, the objective of lines 1–9 is to make a group of trades $\mathtt{\acute{Tr}_j}$ for each session node $\mathtt{N_s^j}$. Hence, $\mathtt{\acute{Tr}}$ is considered a set of sets for all groups of trades. Each $\mathtt{N_s^j}$ is assigned a unique random number $\mathtt{R^j}$ from range $\mathtt{[1, \bar{\bar{N}}_s]}$. Finally, each $\mathtt{\acute{Tr}_j}$ and corresponding $\mathtt{R^j}$ is sent for verification to a randomly selected but unique $\mathtt{N_s^i}$ such that $\mathtt{N_s^i} \neq \mathtt{N_s^j}$. In other words, each node in $\mathtt{N_s}$ receives one and only one $\mathtt{\acute{Tr}_j}$, and it is not the original forwarder for that trade to the orderer. Lines 10–19 ensure that either 51% ($\lceil[(\bar{\bar{\mathtt{N}}}_\mathtt{s})/2]+1\rceil$) of the nodes respond in positive verification or a timeout happens which ensures that block formation does not continue indefinitely. During this time, the verification response messages must contain the assigned random number to ensure that the selected session node is responding. The sum of all random numbers for this session is computed as

$$S^{N_s} = \bar{\bar{N}}_s\left(1 + \frac{\bar{\bar{N}}_s - 1}{2}\right) \tag{1}$$

where, $\bar{\bar{\mathtt{N}}}_\mathtt{s}$ is the cardinality of $\mathtt{N_s}$. While processing verified responses, the algorithm computes the sum of random numbers as $\mathtt{S}_\mathtt{R}^{\mathtt{N}_\mathtt{s}^\mathtt{r}} = \sum_{\mathtt{j=1}}^{[(\bar{\bar{\mathtt{N}}}_\mathtt{s})/2]+1} \mathtt{R}^\mathtt{j}$, where $\mathtt{N}_\mathtt{s}^\mathtt{r} \in \mathtt{N_s}$ is the set of responding nodes. Finally, in lines 20–25, true is returned for block creation if the sum of random numbers from $\mathtt{N}_\mathtt{s}^\mathtt{r}$ and non-negative responding session nodes is equal to $\mathtt{S}^{\mathtt{N_s}}$ given by (1). Line 20 ensures that those nodes, which are *nonresponsive* or have returned incorrect random number (*faulty nodes*) are not counted toward the sum. Hence, only those nodes, which returned correct values contribute to the consensus, and any malicious attempt is thwarted. The orderer only commits the block if Algorithm 2 returns true. If neither true nor false is return within the block creation time, then timeout anomalies occur, which are discussed in Section V-B.

Fig. 4(b) shows the response generated by $\mathtt{N_s}$ along with the random numbers. As the orderer receives $\lceil(6/2)\rceil + 1 = 4$ responses, the random number calculations are performed, and the block is committed if they match. The verification at $\mathtt{N}_\mathtt{s}^\mathtt{i}$ is performed using Algorithm 3. It is important to note that for privacy reasons the verifying nodes cannot read the trade data content [35]. The content is encrypted by the originator of the trade, and the keys are not shared with verifying nodes. They only check the signatures of participating nodes, chaincode validity, and orderer identity. If all of them are found to be valid, then the random number received is returned, otherwise, a negative value is returned. Note that signatures are not part of encrypted content, rather they are metadata in a trade message.

3) Computation Time Calculation: The time required to perform verification for several nodes directly affects the overall transaction rate of the system as well as its reliability. In the proposed PoBT mechanism, the time required is initially based on the individual verification of trade between two nodes, and then verification by $\mathtt{N_s}$ during consensus formation. Hence, the total computation time $\mathtt{T_b}$ can be computed as

$$T_b = \sum_{i=1}^{\bar{\bar{Tr}}} Tr^i(t) + \sum_{i=1}^{\frac{\bar{\bar{N}}_s}{2}+1} N_s^i(t) \tag{2}$$

where, $\mathtt{Tr^i(t)}$ represents time for one trade, and $\mathtt{N_s^i(t)}$ is the time for verification by $\mathtt{N_s}$ during consensus. Similarly, for Hyperledger Fabric the computation time $\mathtt{T_b^{Fab}}$ can be calculated using

$$T_\mathrm{b}^{\mathrm{Fab}} = \prod_{j=1}^{\bar{\bar{Tr}}} \sum_{i=1}^{\bar{\bar{N}}_s} N_s^i(t). \tag{3}$$

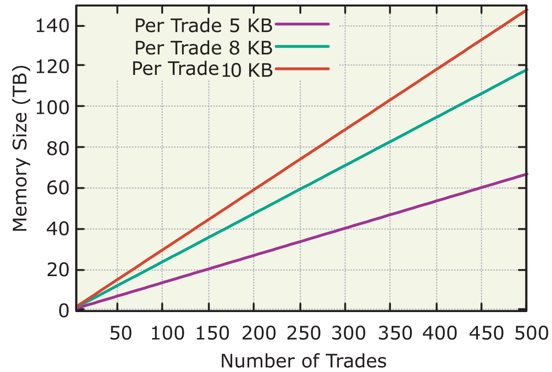


Fig. 5. Ledger memory scalability.

By comparison of (2) and (3), it can be observed that the proposed algorithm will consume less time in the verification process (summation), as compared to the existing state-of-the-art Fabric solution (product), for the same number of trades and endorsing nodes.

V. Local Trades and Timeout Anomalies

There are two major issues with blockchain for IoT in addition to the consensus formation, which are addressed in this section. The first one concerns the memory scalability of the ledger. Each node stores a replica of the ledger and over time the memory requirements become larger. The second challenge is to deal with the block timeouts, where several trades (which may be valid) are not added to the ledger as the consensus cannot be reached in a specified time. This increases the trade failure rate.

A. Ledger Scalability: Local Trade Process

The generic working of blockchain involves distributing a committed block (with all trades in a session) among all the nodes in the network. Hence, the overall memory required for the ledger proportionally increases with the number of nodes in the network. Memory utilization depends on many factors, such as transaction format, storage policy, transaction content, and how frequently blocks are formed. As shown in Fig. 5, with a ten node network, the memory required to store committed blocks increases sharply. For different IoT applications, the size of trades will vary, hence the memory required can increase to hundreds of terabytes (e.g., 10-KB trades compared to 5-KB trades). Although more nodes certainly mean a higher number of validating nodes as well, which implies enhanced security, but keeping in view the size and capabilities of nodes, the memory required to store these trades may become an issue.

Let us denote size of a trade as $\mathtt{Tr_w}$, weight of the block header as $\mathtt{B_w}$, and trades per block as $\bar{\bar{\mathtt{Tr}}}$. The ledger weight $\mathtt{Ld_w}$ (in bytes) will increase for a specific time period according to (4), where $i = 1, 2, 3, \ldots, n$ denotes the number of blocks in a specific time series

$$Ld_w = \sum_{i=1}^{n} \sum_{j=1}^{\bar{\bar{\mathrm{Tr}}}} \mathrm{Tr}_w^j + B_w^i. \tag{4}$$

From here, if the average trade acceptance rate per second is $\mathtt{Tr_r}$, then the ledger increase rate per second can be computed as

$$Ld_{w/s} = \left(\mathrm{Tr}_w + \frac{B_w}{\mathrm{Tr}_n}\right) \times \mathrm{Tr}_r. \tag{5}$$

In practical terms, assume an IoT blockchain handles 10 trades per second. From experimental evaluation on Hyperledger Fabric we know, every single trade is $\approx 5-10$ KB on average, a block is formed with an average of 500 trades, and a block header is 4.5 KB. From (4), the rate at which ledger size increases is computed as $\approx 50-100$ KB/sec or $\approx 4-8$ GB/day or $\approx 1.5-3\text{TB/year}$. Although this does not seem very high for a single node, with hundreds of IoT nodes it becomes impractical.

We address this problem by segregating the trades among devices connected to the same node, from the trades among devices connected to different nodes. At trade origination phase, source node $\mathtt{N_{sd}}$ defines the trade execution path. Fig. 6 depicts the complete process of local trade verification and block formation. Here, $\mathtt{N_{sd}}$ is the local node, and $\mathtt{N_i} \in \mathtt{N}$ is a randomly chosen node. Once $\mathtt{N_{sd}}$ ascertains that both source $\mathtt{D_s}$ and destination $\mathtt{D_d}$ IoT devices are associated to $\mathtt{N_{sd}}$, it verifies their signatures, and then requests the orderer to select a random node $\mathtt{N_i}$ which cross validates the trades. This is the only consensus formation process for such trades. The random selection ensures that the node is different for every session, and a compromised $\mathtt{N_{sd}}$ cannot choose a specific validator. Once verified the trade is forwarded to the orderer from $\mathtt{N_i}$, which ensures that a compromised $\mathtt{N_{sd}}$ has not skipped this process. The orderer ensures the validity of signatures and commits the block afterward. The block is kept at $\mathtt{N_{sd}}$ only, which ensures that the memory of other nodes is not utilized for such trades. The orderer has an internal process to maintain a list of unique block IDs for local and global trades, hence, it can provide the relevant ID whenever desired by any node with proper access rights. The whole process enables three key properties: 1) the verification/consensus is not skipped; 2) communication and computation overhead are reduced; and 3) ledger scalability is significantly increased. Continuing the earlier example, assume internode to intranode trade ratio of 2:3, the size of the distributed ledger is reduced to $\approx 0.5-1.2\text{TB/yr}$, which is more than 50% reduction at each instance of the ledger.

B. Timeout Anomaly: Trade Resubmission

In a generic BBC, the blocks are finalized based on batch timeout which is fixed by the developer. If for any reason (e.g., bandwidth, cyber attack, error, etc.) the orderer failed to form a consensus within due time, an operation timeout error occurs. All trades are rejected and sent back to the sources. This is a current and major challenge in blockchain referred to as multiversion concurrency control [36].

The application should be designed to detect such an anomaly and resubmit the trades without the user's knowledge. Although this increases the complexity of algorithms, it automates the system and reduces unnecessary trade rejection. We deduce that this failure may happen at two levels: 1) early failure (during the preBlocking/consensus process)

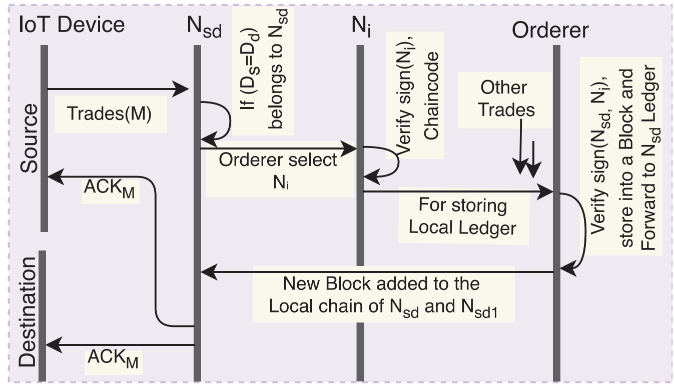


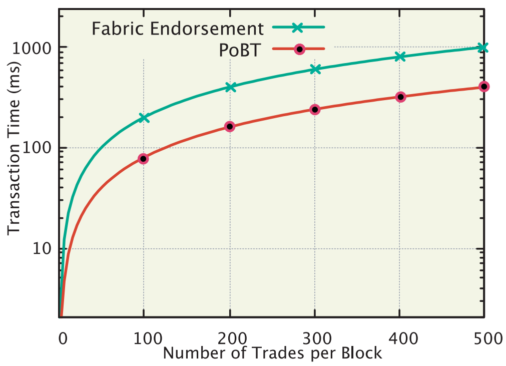
Fig. 6. Local trade execution: operation flow.

and 2) late failure (during the committing phase). In this article, we handle both issues at the orderer. The orderer remains in *active_blocking* state, which ensures that a new block creation is not started (Algorithm 1: line 5). In the commit phase, the verified block is issued by the orderer and the world state is updated as described in Section III-C. The late failure occurs when the world state is not updated (for any reason), hence, commit is attempted again for distribution of block, and *active_blocking* is only removed after the successful commit. An early failure occurs when the block formation time expires but sufficient responses have not been collected to make a decision on successful (or unsuccessful) formation of the block. Hence, the orderer adds the trades from *timeout* block to the next session, and removes *active_blocking* state. If there are trades from the same D_s, the *timeout* trades are rejected and new trades are added. This ensures that double spending is controlled.

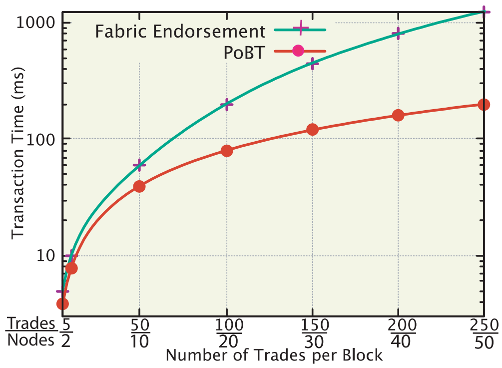
VI. Analysis and Evaluation

In this section, we present the security analysis of the proposed algorithms, followed by the experimental evaluation of consensus formation algorithms compared to the existing Hyperledger Fabric BBC solution. It is important to note that the baseline Fabric is used as a benchmark, so that other works in the future can compare the performance to the proposed solution. Furthermore, as the solution is for BBC, hence cryptocurrency (mining) algorithms cannot be compared in the IoT environment. The evaluation has been done by implementing the proposed consensus algorithm on top of the Hyperledger Fabric v1.0.2. The IoT device trade generation is done through IBM's Node-Red application, which generates concurrent trades fed to nodes running in Docker containers. Each Docker container contains the Fabric *peer-code* along with SCs (chain code library) in GoLang. Two systems with Core-i7 with 2.7-Ghz 16-GB RAM and Core-i5 with 3-GHz 8-GB RAM host the Docker containers. The ordering service is Kafka-based, while the chaincode shim executes the function of getState, getRAngeQuery, and putState for blockchain operations.

Several experiments have been done to evaluate different performance metrics. Hence, the input parameters for the number of trades per block, number of peers, concurrent trades,

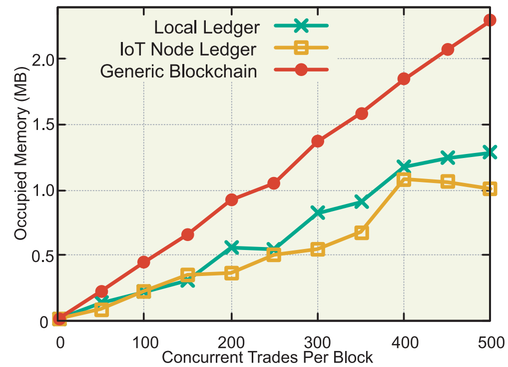


(a)

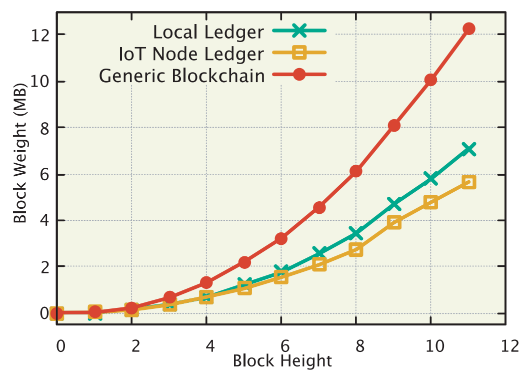


(b)

Fig. 7. Computation time of the PoBT algorithm. (a) Fixed number of nodes. (b) Variable number of nodes and trades.



(a)



(b)

Fig. 8. Memory requirement of local ledger. (a) Memory per block. (b) Ledger memory.

and blocking times, vary in each experiment. These details are given in the following sections with individual experiments. Each experiment is executed 20 times, and the average results are presented.

A. Security Analysis

In a blockchain system designed for IoT applications, there can be two types of adversaries.

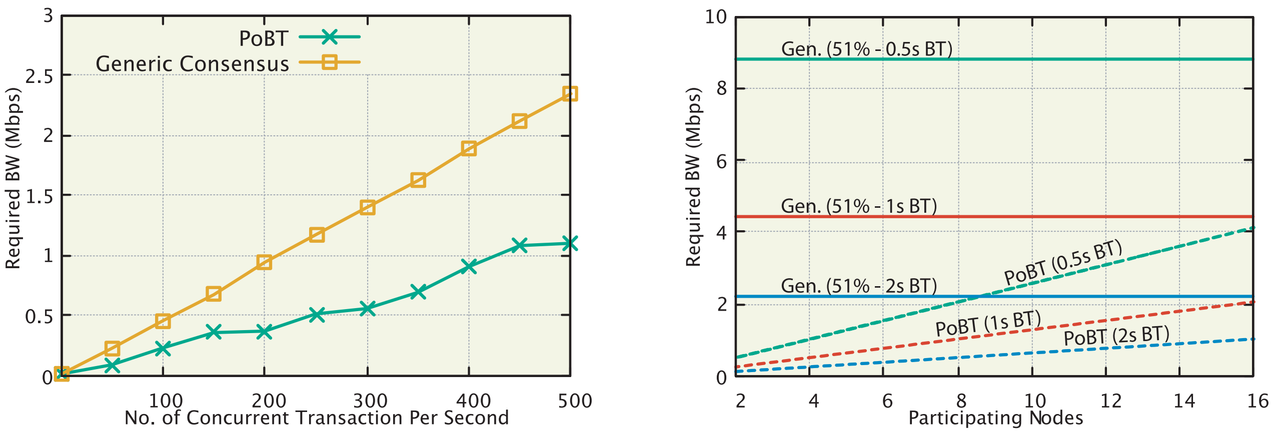
1) *External:* Nonmember devices or nodes may try to become a part of the network, or try to impersonate an existing authenticated entities (including applications [37]).
2) *Internal:* Devices and nodes which are properly registered and have valid signatures may become rogue due to malware or hacking.

In either case, the objective of an attack would be to have an invalid trade endorsed and committed to the ledger.

1) Validation of Individual Trade (Before Block Formation): Once a device initiates a trade, the local node validates the source and destination and the SC. For an external attack, the device will not be able to provide legitimate certificates, hence validation will fail immediately. For an internal attacker, the trade will be valid as long as the SC holds. In Hyperledger Fabric, trades are endorsed by a set of nodes and committed to the ledger. However, in our solution, trade is forwarded to the node responsible for a destination device. It counter validates all credentials and forwards the trade to the order. This eliminates the possibility of rouge source nodes sending the illegal trades to the orderer. Here, even if both $\mathtt{D_s}$ and $\mathtt{D_d}$ along with the respective nodes are compromised, and collude to inject an illegal trade, the orderer will perform consensus to ensure that no illegal trade enters the ledger.

2) Validation of Block (at Block Formation): In order to ensure that no illegal trade is committed all proposed trades are grouped and validated by $\mathtt{N_s}$. Here, $\mathtt{N_s}$ is different for each block depending on the trader involved. Thus, for a rogue node to be included in $\mathtt{N_s}$, it must also have a validated trade sent to the orderer (as discussed previously). However, the candidate block must have validation from at least 51% $\mathtt{N_s}$, where it is impossible to predetermine the nodes involved in any given session.

1) *Formation of a Block by N_{ns} Nodes:* This is virtually impossible, as the candidate block and random numbers are only provided to N_s. Hence, they do not know session creation trades or participants. Even if this information is somehow obtained, $\mathtt{N_{ns}} \geq (\mathtt{N_s}/2) + 1$ must hold. Here, $\mathtt{N_{ns}}$ must be 51% of $\mathtt{N}$, and all must validate the candidate block with a correct random number. The counter check to this is the summation and crosschecking of a random number provided to the respective nodes in N_s by the orderer. Hence, $\mathtt{N_{ns}}$ cannot form a block.
2) *Illegal Formation of Block by N_s Nodes:* In order for $\mathtt{N_s}$ nodes to validate a compromised block, 51% of $\mathtt{N_s}$ must also be compromised. A compromised node will only be



(a) (b)

Fig. 9. Bandwidth requirement at (a) peers and (b) orderer in the blockchain system. (a) Increase in the transaction rate. (b) Block closing time (BT) constraints with different participating nodes.

part of $\mathtt{N_S}$ if it has a trade sent to orderer, which means that it must also have a compromised source node and a compromised SC. It is important to note that compromising a node is not easy. Hence, to successfully launch such an attack with two compromised nodes, it requires $\bar{\bar{\mathtt{N}}}_{\mathtt{S}} = 3$. Considering the scale of IoT devices and corresponding nodes, for $\bar{\bar{\mathtt{N}}}_{\mathtt{S}} \geq 4$ the required number of compromised nodes have to increase significantly. Hence, the attack probability will be considerably low.

B. Time Requirement Analysis

For computation time (from trade validation to block closing) analysis, we compare the required for our proposed PoBT algorithm with Hyperledger Fabric. The total number of nodes in the system is fixed at 10, while the number of trades per block ranges between 1 to 500. Fig. 7(a) presents the results in terms of time required against the increasing number of trades per block. It is important to note that the scale is logarithmic. It can be observed that, when concurrent trades are 100, endorsement requires $\approx$ 200 ms while PoBT requires $\approx$ 80 ms. Similarly, for 300 trades, endorsement time is $\approx$ 600 ms while PoBT requires about 210 ms which is one-third compared to Fabric. As the concurrent trades increase, both endorsement times increase. However, PoBT computation time is significantly less than that of Fabric.

In Fig. 7(b), we present the time required for trade finalization against two different variable conditions. The x-axis shows the number of trades per block (ranging from 5 to 250) and the associated number of consensus participants (ranging from 2 to 50). This is an important factor as the number of endorsers directly impacts system performance. In the first scenario, for 5 trades with 2 endorses per session, the time required is $\approx$ 4 ms and $\approx$ 3.8 ms, which is almost comparable. However, when number endorsers per session are 20 and trades are 100, then endorsing time increases sharply ($\approx$ 200 ms) for Fabric, while PoBT requires only $\approx$ 80 ms. When nodes are 50 for 250 trades, then endorsement time is about 1.25 s while PoBT required only about 200 ms. From this analysis, it can be observed that the PoBT has superior performance in terms of time required to complete the trade finalization. In both the results of Fig. 7, the Fabric's performance has been obtained by configuring it for 51% BFT. With 100% nodes participating in the consensus process (such as the PoET solution) the time required would be significantly higher. Hence, the proposed solution performs better for increasing the number of trades and participating nodes.

C. Memory Requirement Analysis

We use the local ledger to improve the scalability and memory requirements at IoT nodes. It is important to note that the consensus formation algorithms do not impact the memory. It is the trade and block size that affects the scalability. In the proposed novel solution, splitting the ledger reduces the memory requirements on each peer. Fig. 8(a) shows the memory required in MB against the increasing number of concurrent trades (from 1 to 500 per block) during block formation. Fabric is shown as a generic chain that grows linearly. It is important to note that each peer node has to allocate this memory. However, in the proposed solution with PoBT, this requirement is reduced and is represented as an IoT Node ledger. This is a significant reduction in terms of memory across all nodes. The local ledger memory is shown for single nodes local trades and does not affect other nodes.

Fig. 8(b) shows the memory needed as the ledger size grows with the addition of blocks. A distributed ledger requires much less memory overall in the IoT system. With PoBT consensus this distribution is still validated at multiple levels of trade verification and block creation.

D. Bandwidth Requirement Analysis

Fig. 9 shows the bandwidth analysis of the proposed PoBT consensus algorithm. Blockchain scalability is dependent on numerous parameters, and the communication speed of the connecting network is perhaps the most important one. As the consensus has to be formed within a given block closing time to achieve the desired transaction rate, it is important to analyze the required bandwidth. We measure this by varying the number of concurrent trades per second (1 to 500), and participating nodes (2 to 16). Fig. 9(a) shows that the bandwidth demand increases as the number of concurrent transaction increases (indirectly by the increasing number of IoT nodes). In a generic blockchain, this demand is almost linear and proportional to the increase in trades. However, in PoBT, it is also significantly lower by $\approx 50\%$. It is important to note that this required bandwidth is for each peer in the blockchain network. PoBT takes advantage of trade segregation for local transactions, hence, the load on blockchain peer nodes is reduced without compromising the security level.

From an orderer perspective, Fig. 9(b) shows the required bandwidth, if a certain block closing timeout has to be achieved. In order to achieve higher transaction rates, blocks have to be committed quickly, which demands that communication delays are minimal. In a generic blockchain system with practical BFT (PBFT), 51% of peers must form a consensus. In this experiment the total participating nodes are 30, hence, to achieve 0.5 s blocking time, a minimum bandwidth of 8.4 Mb/s is desired at the orderer. However, with PoBT as the number of participating peers is dependent on participating nodes in a block, hence the required bandwidth extremely less, even if 51% of nodes are participating in a session. The higher block closing time would allow more delay in the communication, hence the data rate requirements are also reduced. It is evident from this experiment that PoBT is extremely efficient in the network scalability from a communications perspective also.

VII. Conclusion

Scalability and security are both crucial for IoT blockchain systems. The success of blockchain is primarily based on consensus formation by more than half of the peers for each block. However, in large-scale systems, this translates to decreased transaction rate as the time to form consensus becomes exponentially long. The modern BBC systems like Hyperledger, have solved this challenge by reducing the involved peers and limiting verification to trades only. However, both of these changes can allow malicious trades to be committed, as block verification is not performed and Byzantine Fault Tolerance is not mandatory. We proposed a PoBT algorithm that enables the security of block at both trade validation and block creation phases. Moreover, we utilized a lightweight consensus algorithm that incorporates peers based on the number of nodes participating in a session. This reduces the computational time required by peers and allows for higher transaction rates for resource-constrained IoT devices. By using a distributed peer system for local and global trades, we have reduced the memory needs at IoT nodes. The performance analysis shows that our proposed algorithm also reduces the bandwidth required at the critical points of the network.

References

[1] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7702–7712, Oct. 2019.

[2] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, "Context-aware cloud robotics for material handling in cognitive industrial Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2272–2281, Aug. 2018.

[3] O. Novo, "Scalable access management in IoT using blockchain: A performance evaluation," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4694–4701, Jun. 2019.

[4] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[5] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Nov. 5, 2019. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[6] W. Mougayar, *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. New York, NY, USA: Wiley, 2016.

[7] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 2nd Quart., 2019.

[8] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4650–4659, Jun. 2019.

[9] S. P. Mohanty, V. P. Yanambaka, E. Kougianos, and D. Puthal, "PUFchain: Hardware-assisted blockchain for sustainable simultaneous device and data security in the Internet of Everything (IoE)," *CoRR*, vol. abs/1909.06496, pp. 1–37, Sep. 2019. [Online]. Available: http://arxiv.org/abs/1909.06496

[10] S. Voshmgir, *Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy*. Berlin, Germany: BlockchainHub, 2019.

[11] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Jun. 2017, pp. 557–564.

[12] P. Tasca and C. J. Tessone, "A taxonomy of blockchain technologies: Principles of identification and classification," *Ledger*, vol. 4, pp. 1–39, Feb. 2019. [Online]. Available: http://ledger.pitt.edu/ojs/index.php/ledger/article/view/140

[13] *Hyperledger: Hyperledger Business Blockchain Technology*. Accessed: Nov. 5, 2019. [Online]. Available: https://www.hyperledger.org/projects

[14] *Hydra Chain Consensus*. Accessed: Nov. 5, 2019. [Online]. Available: https://github.com/HydraChain/hydrachain

[15] G. Greenspan, "Multichain private blockchain," London, U.K., Coin Services Ltd., White Paper. Accessed: Nov. 5, 2019. [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf

[16] *Parity Blockchain Technologies*. Accessed: Nov. 5, 2019. [Online]. Available: https://www.parity.io

[17] P. Morgan. *Quorum: Whitepaper*. Accessed: Nov. 5, 2019. [Online]. Available: https://github.com/jpmorganchase/quorum-docs/blob/master/Blockchain-QuorumHyperledger-20160922.pdf

[18] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. *Corda: An Introduction*. Accessed: Nov. 5, 2019. [Online]. Available: https://docs.corda.net/static/corda-introductory-whitepaper.pdf

[19] "BigchainDB 2.0: The blockchain database," Berlin, Germany, BigchainDB, White Paper. Accessed: Nov. 5, 2019. [Online]. Available: https://www.bigchaindb.com/whitepaper/

[20] *Openchain 0.7 Documentation*. Accessed: Nov. 5, 2019. [Online]. Available: https://docs.openchain.org/en/latest/general/overview.html

[21] S. Popov. *The Tangle*. Accessed: Nov. 5, 2019. [Online]. Available: https://assets.ctfassets.net/

[22] *Hyperledger Sawtooth Documentation*. Accessed: Nov. 5, 2019. [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0.4/

[23] *Introduction to Hyperledger, Volume 1: Business Blockchain Design Philosophy and Consensus*. Accessed: Nov. 5, 2019. [Online]. Available: https://www.hyperledger.org/resources/publications

[24] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consum. Electron. Mag.*, vol. 7, no. 4, pp. 6–14, Jul. 2018.

[25] *Hyperledger Iroha—An Overview*. Accessed: Nov. 5, 2019. [Online]. Available: https://hyperledger.github.io/iroha-api/overview

[26] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for Internet-scale systems," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, pp. 1–14.

[27] Y. Wang *et al.*, "Study of blockchains's consensus mechanism based on credit," *IEEE Access*, vol. 7, pp. 10224–10231, 2019.

[28] D. Puthal and S. P. Mohanty, "Proof of authentication: IoT-friendly blockchains," *IEEE Potentials*, vol. 38, no. 1, pp. 26–29, Jan./Feb. 2019.

[29] D. Puthal, S. P. Mohanty, P. Nanda, E. Kougianos, and G. Das, "Proof-of-authentication for scalable blockchain in resource-constrained distributed systems," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–5.

[30] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled Internet of Vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.

[31] J. An, D. Liang, X. Gui, H. Yang, R. Gui, and X. He, "Crowdsensing quality control and grading evaluation based on a two-consensus blockchain," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4711–4718, Jun. 2019.

[32] J. Bae and H. Lim, "Random mining group selection to prevent 51% attacks on bitcoin," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Jun. 2018, pp. 81–82.

[33] M. Vukolić, "Rethinking permissioned blockchains," in *Proc. ACM Workshop Blockchain Cryptocurrencies Contracts*, 2017, pp. 3–7.

[34] A. Baliga, "Understanding blockchain consensus models," Maharashtra, India, Persistent Syst., White Paper. Accessed: Nov. 5, 2019. [Online]. Available: https://www.persistent.com/wp-content/uploads/2018/02/wp-understanding-blockchain-consensus-models.pdf

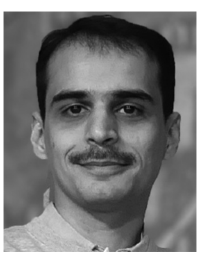
[35] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Netw.*, vol. 32, no. 6, pp. 184–192, Nov./Dec. 2018.

[36] A. O'Dowed, V. Ramakrishna, P. Novotny, N. Gaur, L. Desrosiers, and S. Baset, *Hands-On Blockchain With Hyperledger*. Boston, MA, USA: O'Reilly, 2018.

[37] S. Biswas, K. Sharif, F. Li, and Y. Liu, "3P framework: Customizable permission architecture for mobile applications," in *Proc. 12th Int. Conf. Wireless Algorithms Syst. Appl.*, 2017, pp. 445–456.

Sujit Biswas (M'19) received the M.Sc. degree in computer engineering from Northwestern Polytechnical University, Xi'an, China, in 2015.

He is enrolled as the Ph.D. Fellow with the Beijing Institute of Technology, Beijing, China. He is also an Assistant Professor with the Computer Science and Engineering Department, Faridpur Engineering College, University of Dhaka, Dhaka, Bangladesh. His basic research interest is in Internet of Things, blockchain, and mobile computing security and privacy.



Kashif Sharif (M'08) received the M.S. degree in information technology from the National University of Sciences and Technology, Islamabad, Pakistan, in 2004, and the Ph.D. degree in computing and informatics from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2012.

He is currently an Associate Professor of research with the Beijing Institute of Technology, Beijing, China. His research interests include data centric networks, blockchain and distributed ledger technologies, wireless and sensor networks, software defined networks, and 5G vehicular and UAV networks.

Dr. Sharif also serves as an Associate Editor for IEEE ACCESS.

Fan Li (M'12) received the B.Eng. and M.Eng. degrees in communications and information system from the Huazhong University of Science and Technology, Wuhan, China, in 1998 and 2001, respectively, the M.Eng. degree in electrical engineering from the University of Delaware, Newark, DE, USA, in 2004, and the Ph.D. degree in computer science from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2008.

She is currently a Professor with the School of Computer Science, Beijing Institute of Technology, Beijing, China. Her current research focuses on wireless networks, ad hoc and sensor networks, and mobile computing.

Prof. Li papers have won Best Paper Awards from IEEE MASS in 2013, the IEEE IPCCC in 2013, the ACM MobiHoc in 2014, and the Tsinghua Science and Technology in 2015. She is a member of ACM.

Sabita Maharjan (SM'19) received the Ph.D. degree in networks and distributed systems from the Simula Research Laboratory, University of Oslo, Oslo, Norway, in 2013.

She is currently a Senior Research Scientist with the Simula Metropolitan Center for Digital Engineering, Oslo, and an Associate Professor (adjunct position) with the University of Oslo. She worked as a Research Engineer with the Institute for Infocomm Research, Singapore, in 2010. She was a Visiting Scholar with Zhejiang Univeristy, Hangzhou, China, in 2011, and a Visiting Research Collaborator with the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2012. She was a Postdoctoral Fellow with Simula Research Laboratory from 2014 to 2016. She publishes regularly in prestigious journals in her field, such as the IEEE TRANSACTIONS ON SMART GRID, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *IEEE Communications Magazine*, *IEEE Network Magazine*, *IEEE Wireless Communications Magazine*, and the IEEE INTERNET OF THINGS JOURNAL. Her current research interests include vehicular networks and 5G, network security and resilience, smart grid communications, Internet of Things, machine-to-machine communication, software defined wireless networking, and advanced vehicle safety.

Dr. Maharjan serves/has served in the technical program committee for conferences, including top conferences like IEEE INFOCOM and IEEE IWQoS.

Saraju P. Mohanty (SM'08) received the bachelor's degree (Hons.) in electrical engineering from the Orissa University of Agriculture and Technology, Bhubaneswar, Odisha, in 1995, the master's degree in systems science and automation from the Indian Institute of Science Bangalore, Bengaluru, India, in 1999, and the Ph.D. degree in computer science and engineering from the University of South Florida, Tampa, FL, USA, in 2003.

He is a Professor with the University of North Texas, Denton, TX, USA. His research is in "smart electronic systems" which has been funded by National Science Foundations, Alexandria, VA, USA, Semiconductor Research Corporation, Durham, NC, USA, U.S. Air Force, Washington, DC, USA, IUSSTF, New Delhi, India, and Mission Innovation Global Alliance, India. He has authored 300 research articles, 4 books, and invented 4 U.S. patents. His has Google Scholar citations with an H-index of 32 and i10-index of 110.

Prof. Mohanty was a recipient of nine Best Paper Awards, the IEEE-CS-TCVLSI Distinguished Leadership Award in 2018 for services to the IEEE and to the VLSI research community, and the 2016 PROSE Award for Best Textbook in Physical Sciences and Mathematics category from the Association of American Publishers for his Mixed-Signal System Design book published by McGraw-Hill. He has delivered 8 keynotes and served on 5 panels at various International Conferences. He has been serving on the Editorial Board of several peer-reviewed international journals, including the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS and the IEEE TRANSACTIONS ON BIG DATA. He is currently the Editor-in-Chief of *IEEE Consumer Electronics Magazine*.

Yu Wang (F'18) received the B.Eng. and M.Eng. degrees in computer science from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the Illinois Institute of Technology, Chicago, IL, USA.

He is a Professor of computer science with the Temple University, Philadelphia, PA, USA. His research has been continuously supported by federal agencies, including U.S. National Science Foundation, Alexandria, VA, USA, U.S. Department of Transportation, Washington, DC, USA, and National Natural Science Foundation of China, Beijing. He has authored and coauthored more than 150 papers in peer reviewed journals and conferences. His research interests include wireless networks, mobile social networks, smart sensing, mobile computing, and algorithm design.

Prof. Wang recipient of four Best Paper Awards, the Ralph E. Powe Junior Faculty Enhancement Awards from Oak Ridge Associated Universities in 2006, the Outstanding Faculty Research Award from College of Computing and Informatics at UNC Charlotte in 2008, and the Overseas Young Scholars Cooperation Research Fund from NSFC in 2014. He has served as the Editorial Board Member of several international journals including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He is a Senior Member of ACM.