Ad Hoc Networks 133 (2022) 102882

Contents lists available at ScienceDirect

Ad Hoc Networks

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

Vericom: A Verification and Communication architecture for IoT-based blockchain

Ali Dorri [a,*], Shailesh Mishra [b], Raja Jurdak [a]

[a] Trusted Networks Lab, School of Computer Science, QUT, Australia
[b] Department of Electrical Engineering, IIT Kharagpur, India

ARTICLE INFO

Keywords:
Blockchain
Traffic management
Scalability

ABSTRACT

Blockchain has received tremendous attention as a secure, distributed, and anonymous framework for the Internet of Things (IoT). As a distributed system, blockchain trades off scalability for distribution, which limits the technology's adaptation for large scale networks such as IoT. All transactions and blocks must be broadcast and verified by all participants which limits scalability and incurs computational and communication overheads. The existing solutions to scale blockchains have so far led to partial recentralization, limiting the technology's original appeal. In this paper, we introduce a distributed yet scalable Verification and Communication architecture for blockchain referred to as Vericom. Vericom concurrently achieves high scalability and distribution using hash function outputs to shift blockchains from broadcast to multicast communication. Unlike conventional blockchains where all nodes must verify new transactions/blocks, Vericom uses the hash of IoT traffic to randomly select a set of nodes to verify transactions/blocks which in turn reduces the processing overhead. Vericom incorporates two layers: (i) transmission layer where a randomized multicasting method is introduced along with a backbone network to route traffic, i.e., transactions and blocks, from the source to the destination, and (ii) verification layer where a set of randomly selected nodes are allocated to verify each transaction or block. The performance evaluation shows that Vericom reduces the packet and processing overhead as compared with conventional blockchains. In the worst case, packet overhead in Vericom scales linearly with the number of nodes while the processing overhead remains scale-independent.

1. Introduction

The Internet of Things (IoT) is a network of millions of low-resource devices that collect and exchange information about the physical environments which is then processed by service providers (SPs) to offer personalized services to the users. Conventional IoT ecosystems rely on a brokered communication model where the communications, authentication, and authorizations are conducted by a central trusted authority. In many situations, geographically proximate IoT devices still have to go through a remote central server to access services which is unlikely to scale when millions of nodes are connected. SPs collect a huge volume of personalized information about the users and thus can build a virtual profile about them which risks user privacy. The conventional security architectures are not directly applicable in IoT as IoT encompasses heterogeneous low-resource devices which come with no or limited built-in security features [1,2]. Most of the existing IoT-specific security solutions largely rely on centralized communication models which suffer from lack of scalability and single point of failure [3–5].

1.1. Motivation

In recent years, blockchain has received tremendous attention to address the outlined challenges in IoT due to its salient features including decentralization, anonymity, trust, and security [6–9]. Blockchain is an immutable database shared across all participating nodes in the network and was first introduced in Bitcoin [10], the first cryptocurrency, in 2008. A transaction represents the basic communication primitive between the participating nodes which is sealed using asymmetric encryption. Blockchain participants are known by a unique Public Key (PK) that can be changed for each transaction which in turn introduces a level of anonymity. All transactions are broadcast in the network and verified by all participants. Transaction verification typically involves matching the PK with the associated signature (both are stored in the transaction). Some nodes, known as validators, may choose to store new transactions in blockchain in the form of a block which requires following a consensus algorithm. The latter protects blockchain security against malicious validators that may attempt to

[*] Corresponding author.
E-mail address: Ali.dorri@qut.edu.au (A. Dorri).

https://doi.org/10.1016/j.adhoc.2022.102882
Received 3 June 2021; Received in revised form 23 April 2022; Accepted 26 April 2022
Available online 6 May 2022
1570-8705/© 2022 Elsevier B.V. All rights reserved.

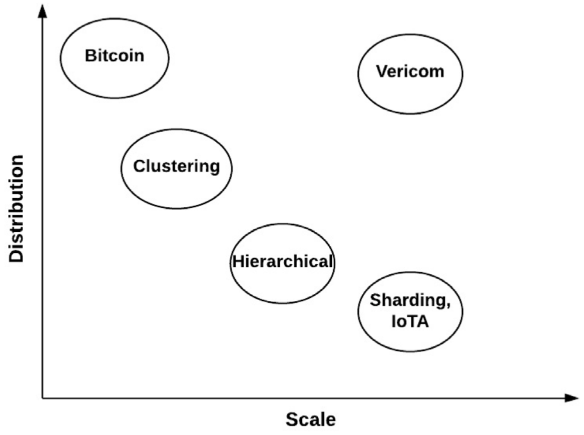


Fig. 1. The traditional correlation between distribution and scale for IoT-based blockchain.

flood the network with fake blocks and ensures the validator of the next block is selected randomly. This introduces a trusted network where untrusted participants can exchange information.

Despite their significant advantages, conventional blockchains are not directly applicable in IoT due to the lack of scalability resulting from huge communication and computational overheads. New transactions and blocks are broadcast to all IoT nodes, i.e., IoT devices and users, to ensure distributed management of the blockchain. This in turn demands significant bandwidth which is far beyond the limited energy, processing, and communication capabilities of the IoT nodes. IoT nodes have limited energy resources. Packet transmission is among the most energy consuming tasks in IoT nodes which makes it impossible for them to participate in the management of the blockchain, where the participants are expected to receive a huge volume of transactions/blocks. New transactions/blocks must be verified by all nodes which in turn requires significant computational resources that is far beyond the capabilities of IoT nodes.

There is a known trade-off between scalability and distribution as shown in Fig. 1. Purely distributed chains, such as Bitcoin [10], suffer from lack of scalability due to their reliance on broadcast communication and computationally demanding block and transaction verification processes where all transactions and blocks needs to be verified by all the participating nodes. Existing proposals to enhance blockchain scalability include hierarchical, sharded, or clustered blockchains (see Fig. 2) [11–13]. In hierarchical methods [13,14], multiple levels of hierarchy are created where the transactions and blocks in each hierarchy are only broadcast to the nodes at the same level in the hierarchy (see Fig. 2a). In sharding [15], the network is divided into multiple groups, i.e., shards, where the transactions of each shard are only broadcast to the nodes in the same shard (see Fig. 2b). In clustering algorithms, the network is grouped into multiple clusters where a high resource available node, known as cluster head (CH), forwards blocks and transactions to/from the cluster members (see Fig. 2c). Unlike sharding where each shard functions independently, in clustering the transactions and blocks are broadcast and the CHs jointly manage the blockchain. In addition to the conventional blockchains, IoTA, a distributed ledger technology, has been introduced to eliminate centralization in IoT. However, IoTA still relies on broadcast communications and suffers from centralization of a coordinator node that verifies transactions.

As shown in Fig. 1 the existing methods sacrifice distribution for scale by deviating from blockchain's original distributed topology. The security and anonymity of the blockchains are directly impacted by the distribution and scaling features. A distributed large scale network achieves higher security as compared with more centralized smaller scale chains as a larger number of potential validators exist in the network, making it harder for attackers to dominate the blockchain network and store fake blocks. Additionally, blocks are verified by a broader set of nodes which protects against colluding nodes that may mark a fake block as valid. From the anonymity perspective, the large number of participants continuously increases the number of transactions and PKs in the blockchain which in turn complicates user deanonymization [16] by linking a group of transactions or PKs to a particular IoT node. Evident from the above discussions, it is critical to reduce the blockchain communication and computational overheads without sacrificing the distributed feature.

1.2. Contributions

The main contribution of this paper is to push the boundaries of distribution and scale (see Fig. 1) to achieve a distributed yet scalable blockchain which is adoptable in IoT ecosystem by introducing a verification and communication architecture known as *Vericom*. Vericom incorporates two layers namely:

(i) Transmission layer: In this layer, we introduce a multicasting algorithm that sends traffic, i.e., transactions and blocks, to a group of IoT nodes that are selected randomly, dynamically, and in an unpredictable manner based on the hash of the traffic. This shift from broadcast (as in conventional blockchains) to multicast significantly reduces the bandwidth consumption of the underlying IoT nodes while the randomness, uniqueness, and dynamic allocation of the destination group avoids centralization. Vericom incorporates a backbone network that receives traffic from the source IoT node and employs IP-based routing algorithms to route the traffic toward the destination IoT nodes that are randomly selected based on the hash of the transaction/block (details in verification layer). The backbone network is similar to the Internet backbone network that routes the Internet traffic even in the existing blockchain architectures. To incentivize nodes to function as backbone nodes, Vericom introduces a traffic management fee (TMF) that is paid by the validators to the backbone nodes. The value of TMF is defined based on the number of blocks generated by a validator during an epoch time.

(ii) Verification layer: In this layer a subset of the participating nodes, known as verifier set, are randomly and dynamically dedicated to verify a particular transaction or block which in turn reduces the computational overhead associated with transaction/block verification on the IoT devices as compared with conventional blockchains where all transactions and blocks are broadcast. The main aim of this layer is to limit the verifiers of the traffic while preserving the security of the ledger against malicious nodes that may mark an invalid block or transaction as valid to store fake data in the blockchain. The selection of the verifier set is based on the hash of the transaction/block content that ensures randomness and unpredictability of the verifier set. This also ensures the distributed nature of Vericom as for each transaction/block a unique verifier set is randomly and dynamically identified. Once a block is verified by the verifier set, it is broadcast to the network and any other node may also attempt to verify the same to protect against rare cases where all the nodes in the verifier set are colluding malicious nodes.

We study the security of Vericom against three possible attacks and discuss how Vericom is resilient against such attacks. The simulation results show that Vericom reduces the packet and computational overhead as compared with conventional blockchains.

1.3. Paper organization

The rest of the paper is organized as follows. Section 2 studies the existing solutions to enhance the blockchain scalability. Section 3 discusses the preliminary concepts upon which Vericom builds. Section 4 outlines the details of Vericom. Section 5 presents the evaluation results and discusses the performance of Vericom and finally Section 6 concludes the paper and outlines future research directions.

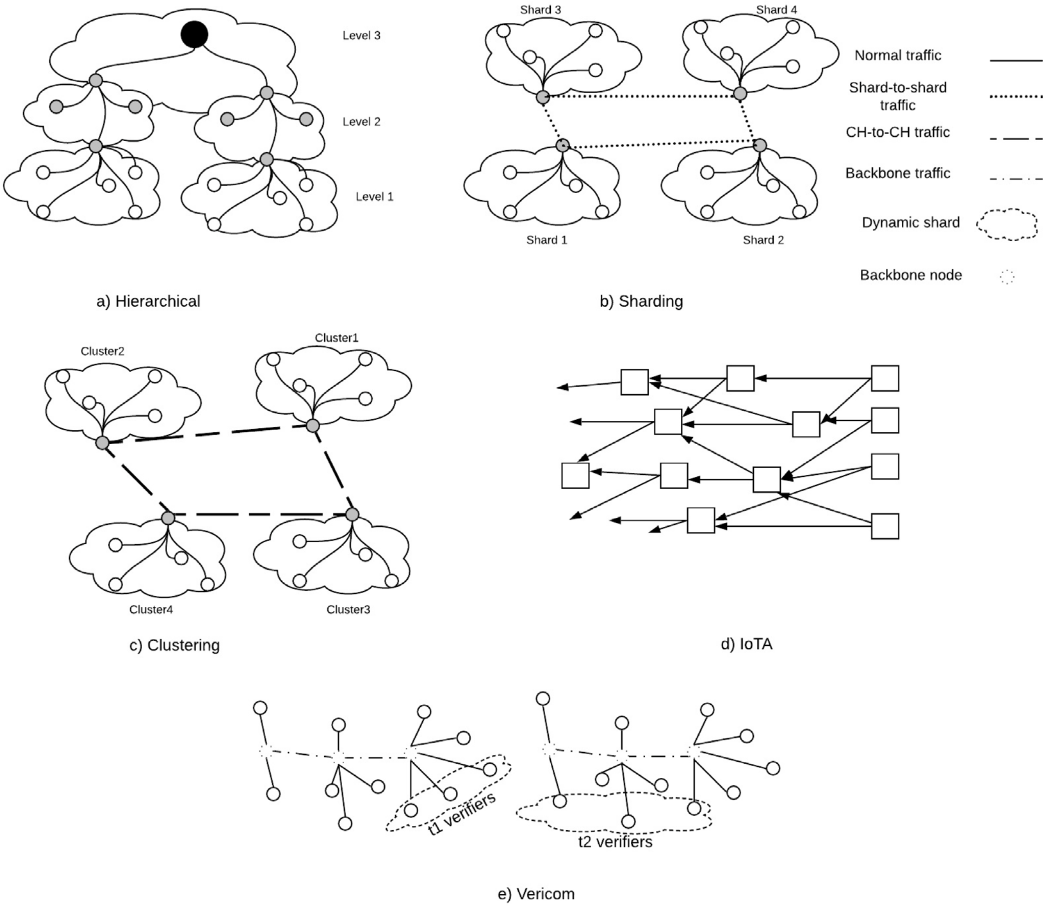


Fig. 2. A high-level topological view of the studied categorizes. (a),(b),(c) create static groups with large number of participants to limit the nodes in the blockchain while (d) represents IoTA where transactions are linked (e) forms dynamic group with small participants that verify transactions.

2. On the scalability of blockchain

In this section, we study the existing solutions in the literature to enhance the blockchain scalability for IoT. Let us first explain the blockchain scalability limitation in a smart grid setting, as a representative use case of IoT, to motivate the discussions in the rest of this section. The penetration of Distributed Energy Resources (DER) and smart meters leads to the emergence of energy prosumers which are the nodes capable of consuming and producing energy. A smart home equipped with solar panels is an example of an energy prosumer. The prosumers may trade their excess energy with other energy consumers which leads to a significant volume of transactions as trading energy involves broadcasting multiple transactions to find consumers/producers and to negotiate trade terms with the intended seller/buyer [17]. Additionally, the grid participants may need to frequently (in 1 h or 0.5 h intervals) send data to the grid operator, or other trusted parties, to ensure the reliability of the grid and balance the energy demand with supply. The transactions and blocks need to be verified by the blockchain participants which in turn demands significant computational resources from the participants while smart grid participants, e.g. solar panels and smart meters, have limited resources.

Having motivated the need for blockchain scalability in cyberphysical environments, we next study the existing optimization methods in the literature to enhance blockchain scalability. Based on the communication topology employed by each optimization method, we classify those in three categories namely hierarchical methods, sharding, and clustering. A high-level topological view of these categories is shown in Fig. 2.

2.1. Hierarchical methods

In hierarchical methods, as shown in Fig. 2a, the network is organized in the form of multiple hierarchies where the transactions in each level of the hierarchy are only broadcast to the nodes in the same level which in turn enhances the blockchain scalability. In each level, a manager node functions as the central authority that authorizes nodes to join the network at that level and forwards traffic to/from the upper level hierarchies. The authors in [11] proposed a hierarchical architecture for access control management in blockchain. The framework comprises three main layers which are device, fog, and cloud layers. The device layer encompasses the IoT devices. The fog layer includes the first tier of the blockchain that is managed by a central node. Each device in the device layer associates with a node in the fog layer that stores the transaction of the device in a private chain. The cloud layer comprises central servers that run the global blockchain and store a copy of the private blockchains.

The authors in [18] proposed a hierarchical architecture that comprises four chains namely: (1) payment engine that handles the payments and micropayments, (2) compute engine that governs the smart contracts and runs distributed applications, (3) storage engine that stores the data associated with transactions, and (4) core engine that comprises of a public ledger that connects all chains and enables transaction exchange between other chains, i.e., engines.

Side chains are introduced to enhance the blockchain scalability and reduce delay in trading assets [19,20]. To create a side chain, a user locks a particular amount of asset in the main chain and transfers it to the side chain where it can be traded with other parties without requiring the transactions to be sent to the main chain. Once the trade is concluded, the asset ownership information is transferred to the parent chain and the side chain is closed [20].

2.2. Sharding

Sharding refers to partitioning the network where each partition, also known as shard, functions independently and is managed by a shard manager (see Fig. 2b). The transactions and blocks generated by the nodes in each shard are only broadcast and verified in the same shard which in turn increases scalability. Shard-to-shard communication is limited to where the verification of a transaction in one shard requires input from a transaction in another shard, e.g., spending the output of a transaction in a different shard. In such cases, the managers of the involved shards communicate to verify the transaction.

The authors in [12] proposed a sharded blockchain architecture to enhance scalability of the blockchain for IoT applications. In each shard a manager node authorizes the nodes that can join the shard, verifies new transactions, and stores new blocks. The shards are connected through a main shard where the shard managers connect to reach consensus on sub-blocks, i.e., the chains in each shard.

2.3. Clustering

Clustering enhances blockchain scalability for IoT by reducing the number of nodes that participate in blockchain management. The network is clustered into multiple groups (see Fig. 2c). In each cluster, a node with sufficient computational resources is selected as the cluster head (CH) that (i) receives transactions from the cluster members and broadcasts to the blockchain, (ii) participates in the blockchain by verifying new transactions and storing blocks, and (iii) forwards transactions to the cluster members if they are the destination.

Unlike sharding where communication between shard managers is limited to selected transactions, clustering broadcasts all blocks and transactions between the CHs. In [13] we introduced a scalable blockchain where new blocks and transactions are broadcast and verified only by the CHs. The cluster members populate an Access Control List (ACL) to authorize particular nodes in the network to send them transactions. CHs employ the ACL to decide whether to send a transaction to the cluster members or to other CHs. In [21] the authors introduced a novel solution to group the transactions and thus participating nodes in the smart grid utilizing smart contracts. Benefiting from the autonomous feature of the smart contract, the grouping is performed without reliance on any third party.

2.4. IoTA

IoTA [22] has been introduced as a scalable solution that can address the limitations of blockchain for large scale IoT. IoTA is not a blockchain but a distributed ledger technology, where unlike blockchains transactions are not committed in the form of blocks. Instead, IoTA introduced the *Tangle* (see Fig. 2d) that creates a directed acyclic graph (DAG) to store transactions. In order for a transaction to be stored in IoTA, the transaction generator must randomly select and verify two previously generated transactions. As more nodes verify a transaction, and thus more transactions are chained to it, the weight and thus confirmation level of the transaction increases.

2.5. Discussion

Having discussed the key methods employed to increase blockchain scalability, we next evaluate such methods and highlight their limitations for IoT.

In the outlined methods, the network is divided into subgroups which are managed by a central manager which in turn leads to a decentralized topology. The manager still suffers from the centralization challenges including privacy, security, single point of failure and scale. The reduced number of participants in each group, facilitates user deanonymization as the malicious nodes deanonymize the users from a smaller pool. In the above methods, there is a trade-off between the number of groups, e.g., clusters, and the centralization degree. Fewer groups increase scalability but lead to more centralization.

In hierarchical methods, the packets will eventually traverse multiple tiers before being considered as valid which in turn increases the delay in verifying transactions. In sharding methods, intra-shard communication and verification remain challenging and incur significant delay.

IoTA relies on broadcast communications which in turn suffers from high packet overhead as in conventional blockchains. IoTA improves the verification processing overhead as fewer nodes confirm a transaction, however, it comes with the cost of increased delay as transactions shall wait for longer time to receive enough weight. Additionally, IoTA relies on a coordinator node that centrally confirms transactions each two minutes which in turn suffers from centralization and moves away from distributed technology [23].

Vericom introduces a distributed yet scalable blockchain by multicasting traffic to a randomly selected set of nodes that verify a transaction/block (see Fig. 2e). The verifier set is unique for each transaction/block and is selected randomly and dynamically based on the hash of the transaction/block content which in turn increases blockchain security against malicious nodes that may store fake transactions. The verifier set dynamically changes per transaction/block which moves away from centralization and thus mitigates the related challenges such as privacy and security. Any of the participating nodes in blockchain may also choose to verify new blocks or transactions to detect misbehavior. Malicious nodes are isolated from the network to mitigate the impact of attacks. By introducing a distributed yet scalable architecture, Vericom supports a larger number of participating nodes which in turn reduces the chance of user deanonymization as a huge volume of transactions are stored in the main chain.

Depending on the read/write permissions of the participating nodes, blockchain can be categorized as [24]: (i) *Public* where all nodes have equal read/write permissions and any node can participate on blockchain, and (ii) *Private* where authorized nodes may only have write permission, i.e., store new blocks and a selected node authorizes the nodes that can participate on the network. The discussion in the rest of this paper applies to both public and private chains, however, the maximum benefit is for the public chains due to the large scale and openness of such ledgers.

Having discussed the motivations behind Vericom and the core contributions, we next study the preliminaries.

3. Preliminaries

In this section, we briefly outline some algorithms that are partially employed in Vericom. Vericom delivers a distributed yet scalable blockchain architecture for IoT by significantly reducing the communication overhead, achieved through dynamic multicast, and the computational overhead, achieved through the use of hash function outputs for randomized and secure verifier set selection. Vericom is part of a larger project with the aim of designing an IoT-friendly blockchain architecture. In our earlier work, Tree-chain [25,26], we introduced a fast consensus algorithm where a leader is selected randomly to commit transactions that match a particular pattern in the blockchain. Tree-chain is a lightweight validator selection algorithm (also known as consensus algorithm in the literature) that significantly reduces the computational overhead and delay associated with storing new blocks by randomizing the selection of validator sets based on hash function output. The upper-bound throughput of the Tree-chain is the speed at which a validator can verify a transaction which is in near real-time. Such fast transaction commitment requires fast transaction delivery to the validators which is impossible in the existing blockchains due to the broadcast nature. Each validator has limited bandwidth which limits

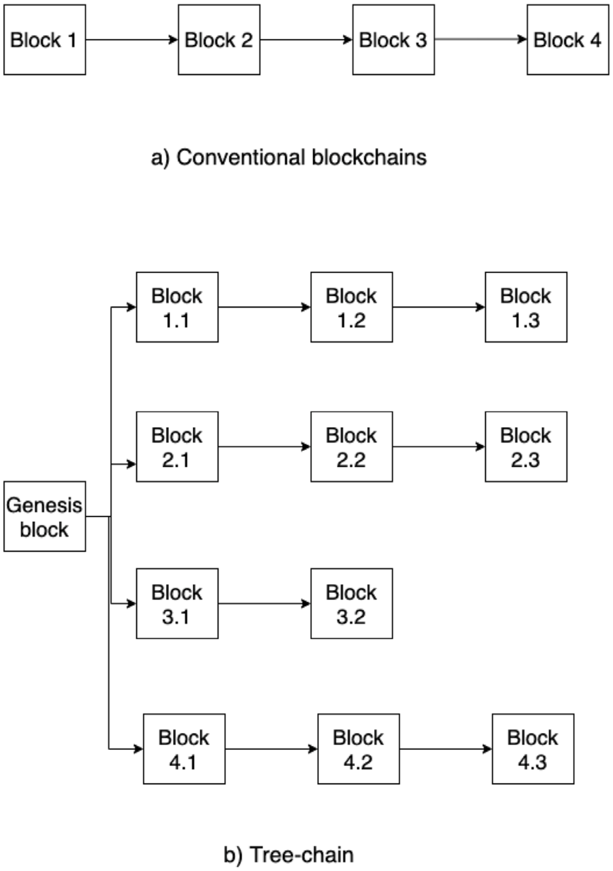


Fig. 3. A high level view of (a) conventional blockchains and (b) Tree-chain.

the number of transactions/blocks it can receive and thus impacts the blockchain throughput. In other words, Tree-chain shifts the transaction throughput bottleneck from the consensus algorithm to the packet propagation and transaction delivery. Vericom aims to address this limitation. In this paper and without loss of generality, we assume that Tree-chain is employed as the underlying validator selection algorithm. However, Vericom is applicable with any other validator selection algorithm.

In Tree-chain the randomization among validators is achieved at two levels:

(i) Transaction level where the validator of each transaction is selected randomly based on the hash of the transaction content. Each validator commits transactions whose hash value starts with a particular character, known as the consensus code.

(ii) Block level where the consensus code corresponding to each validator is randomly allocated based on the hash of the PK of the validator.

As shown in Fig. 3, Tree-chain embraces the concept of forking where each ledger is managed by a particular validator, i.e., each ledger contains transactions that all fall within the same consensus code. Depending on the weight of their PK, the validators are allocated a *Consensus Code Range*, which is the most significant characters of the hash function output. Each validator only commits transactions whose hash falls within its corresponding consensus code range. Unlike tree-chain, Vericom is not a validator selection algorithm. Instead, Vericom focuses on routing and verifying the already generated blocks and transactions.

Another basic building block we partially employed in Vericom is the routing algorithm proposed in our earlier work [27]. We introduced an anonymous routing method that routes transactions from the source node to the destination on the basis of PK. Designated nodes in the network form a backbone network. Each backbone node receives transactions where the hash of the transaction destination starts with specific characters. The destination nodes join the backbone node that is responsible for managing their transactions by sending a join request transaction to the backbone node. The backbone nodes route transactions based on the PK of the destination.

Table 1
Definition of the abbreviations and indexes used in this paper.

Notion	Meaning
PK	Public Key
CCR	Consensus Code Range
VN	Validator Node
NN	Normal Node
BN	Backbone Node
RUI	Route Update Interval
TMF	Traffic Management Fee
TF	Traffic Fee
TA	Traffic Accounting
WD	Weight Dictionary
VRD	Validation Range Distributor
VR	Validation Range
KWM	Key Weight Metric

In this paper, we employ the concept of backbone network, however, the backbone network is responsible for managing all blockchain traffic, including blocks and transactions, and introduces a new method to define the destination of the incoming traffic based the hash function output of the traffic. The backbone nodes do not broadcast the traffic, instead they multicast the traffic flow to a verifier set, which in turn reduces the packet overhead and bandwidth consumption.

Having discussed the background information, we next outline the details of Vericom.

4. Vericom: A verification and communication architecture for IoT-based blockchain

This section outlines the details of Vericom. Table 1 presents a definition of the key abbreviations used in this paper.

4.1. Overview

Vericom introduces a distributed yet scalable blockchain by shifting from broadcasting to dynamic and randomized multicasting of the traffic flow. A high-level picture of Vericom is shown in Fig. 5 and a summary of the core steps is given in Algorithm 1. As Vericom logic is distributed across several network entities, the first entity mentioned in each line in Algorithm 1 is the entity that conducts the action explained in the line. The verifiers of a particular transaction/block are identified based on the output of the hash of the traffic content which ensures randomized and dynamic verifier selection. This in turn reduces the communications overhead and computational overhead for verifying new blocks and transactions. Verifying a transaction/block typically involves matching the PK with the corresponding signature. Vericom consists of two layers namely:

- transmission layer: where the traffic is multicasted to a dynamically selected group of recipients by a group of nodes selected as backbone nodes. The latter are nodes with higher computational resources that route traffic based on the hash of the traffic content. In conventional blockchains, the packets eventually are routed by the Internet backbone network using IP. Vericom incorporates the backbone network which reduces packet overhead and delay in transmitting data as traffic is directly routed by the backbone network rather than broadcast to all nodes. This in turn reduces the number of hops traffic needs to travel,

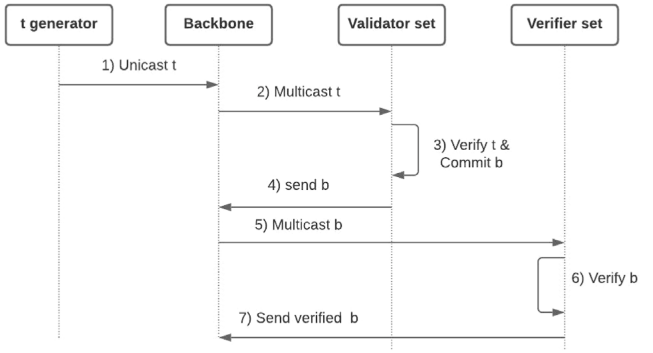


Fig. 4. A high level view of information flow in Vericom (*t* refers to a transaction and *b* refers to a block).

Algorithm 1 A summary of Vericom (N: normal node, BN: backbone node, VN: verifier node).

```
 1: Input: t_n
 2: Output: Verified block
 3: N: Send $t_n$ to BN                                  ▷ Transmission layer
 4: BN: Identify the validator based on hash
 5: BN: Multicast $t_n$ to VN
 6: if VN matches $t_n$.PK with $t_n$.Sign then            ▷ Verification layer
 7:     $t_n$ is verified
 8: VN: Send $t_n$ to BN                                 ▷ Transmission layer
 9: BN: Send $t_n$ to the Validator (Val)
10: Val: Commit $t_n$ to a new block (B)
11: Val: Send B to BN                                    ▷ Transmission layer
12: BN: Multicast B to the verifiers
13: VN: Verify B                                         ▷ Verification layer
14: VN: Send verified B to BN                            ▷ Transmission layer
15: BN: Broadcast B
```

Table 2
Consensus code allocation for scenario in Fig. 5.

Validator ID	Consensus code
VN.1	[0–9]
VN.2	[A–H]
VN.3	[I–P]
VN.4	[Q–Z]

- verification layer: where a set of nodes are randomly selected to verify transactions or blocks. Vericom differentiates between verifiers and validators. A verifier is a node that verifies a newly generated block, while validator refers to a node that first verifies newly generated transactions then commits them in the blockchain by following the Tree-chain consensus algorithm. By reducing the number of nodes that need to verify traffic the computational overhead significantly reduces.

To protect the security of the framework against malicious nodes that may falsely claim a fake transaction as valid, Vericom selects the verifier/validator set randomly based on the hash of the traffic content. The dynamicity and unpredictability of this process are fundamental in ensuring the security of Vericom while reducing overheads. Fig. 4 depicts a high-level view of Vericom traffic flow. Algorithm 1 identifies the flow in which the traffic passes these layers which is further explained in the rest of this section.

As an example scenario, consider the network shown in Fig. 5. Backbone Node.1 (BN.1), BN.2, BN.3, and BN.4 form the backbone network. Assume the consensus code allocated to each of the Validator Nodes (VNs) is as in Table 2. Normal Node.1 (NN.1) generates a transaction whose hash is “K23HQ” and sends to its corresponding backbone node, i.e. BN.1 (line 3, Algorithm 1). The transaction is multicasted to: (i) the validator set (line 4) which in this example are the main validator (selected based on the consensus code allocations and the hash of the transaction, VN.3 in this scenario, and (ii) the validator that is allocated to the next consensus code range of the main validator, i.e., VN.4 in this scenario. BN.1 routes the transaction to BN.3 and BN.4 using their corresponding IP addresses (line 5). Each BN populates and updates a routing table based on conventional IP-based routing algorithms that is used to route traffic in backbone network. Upon receipt of the transaction, BN.3 and BN.4 send the transaction to VN.3 and VN.4 to be verified (lines 6&7) and committed in the blockchain. Depending on the application, the verification of a transaction may involve different steps. Recall that we assume Tree-chain is employed as the underlying consensus algorithm, thus the main validator commits the transaction in the blockchain (line 10, the validator skips lines 8&9 as the same nodes that verify transactions commit them in blockchain). After generating a new block, the main validator sends the block to the backbone node (line 11) which is then multicasted to the verifier set (line 12). After verifying the new block, the verifiers send it to the backbone nodes to be broadcast in the network (lines13–15). In conventional blockchains, for a transaction to be verified and committed, two broadcasts shall happen (broadcasting the transaction and the block). Vericom relies on multicasting for transaction and block verification, however, eventually the block needs to be broadcast to be stored by the nodes. The storage process is beyond the scope of Vericom and thus we leave that for future work. We discuss the details of the outlined process in the rest of this section.

4.2. Transmission layer

Unlike conventional blockchains where traffic is broadcast, in Vericom, the traffic is routed by the backbone nodes and then multicasted to intended recipient groups that are dynamically identified based on the verification layer (discussed later in this section). To achieve this goal, we introduce a backbone network along with a PK-based multicasting. Highly stable and resourceful nodes in the network jointly form a *backbone* network that is the core for traffic management. The backbone network receives traffic from all IoT nodes, e.g. solar panels or IoT devices in a smart home, and delivers it to the intended destination. The formation of the backbone depends on the level of trust to the backbone nodes. We consider two scenarios, trusted backbone nodes and untrusted backbone nodes, which we discuss separately below.

4.2.1. Trusted backbone nodes

In this scenario, the backbone nodes are trusted, e.g., the servers provided by the Internet Service Providers (ISPs) or the government. This is similar to the Internet backbone network that manages the main Internet traffic flow. In conventional blockchains the traffic is technically broadcast in an overlay network, while in the lower layers, the traffic is routed by the *trusted* Internet backbone nodes (as with any other Internet traffic). Vericom aims to remove the overlay network in conventional blockchains and incorporate the backbone network concept in the blockchain design to reduce delay and packet overhead. As outlined earlier, even the existing pure distributed blockchains rely on trusted Internet backbone nodes to relay traffic, thus, the trust level to the backbone nodes does not impact the distributed nature of the blockchain. The reason is that while communication traverses the backbone, the routing decisions are made independently of the backbone. Conventionally, transactions and blocks are broadcast to all participants. Vericom replaces broadcast with dynamic multicast to a randomly selected set of nodes based on hash outputs, which maintains independence of routing decisions from the backbone and avoids any centralization of trust. Apart from the underlying transmission layer, blockchain tasks are still conducted in a distributed manner as outlined later in verification layer.

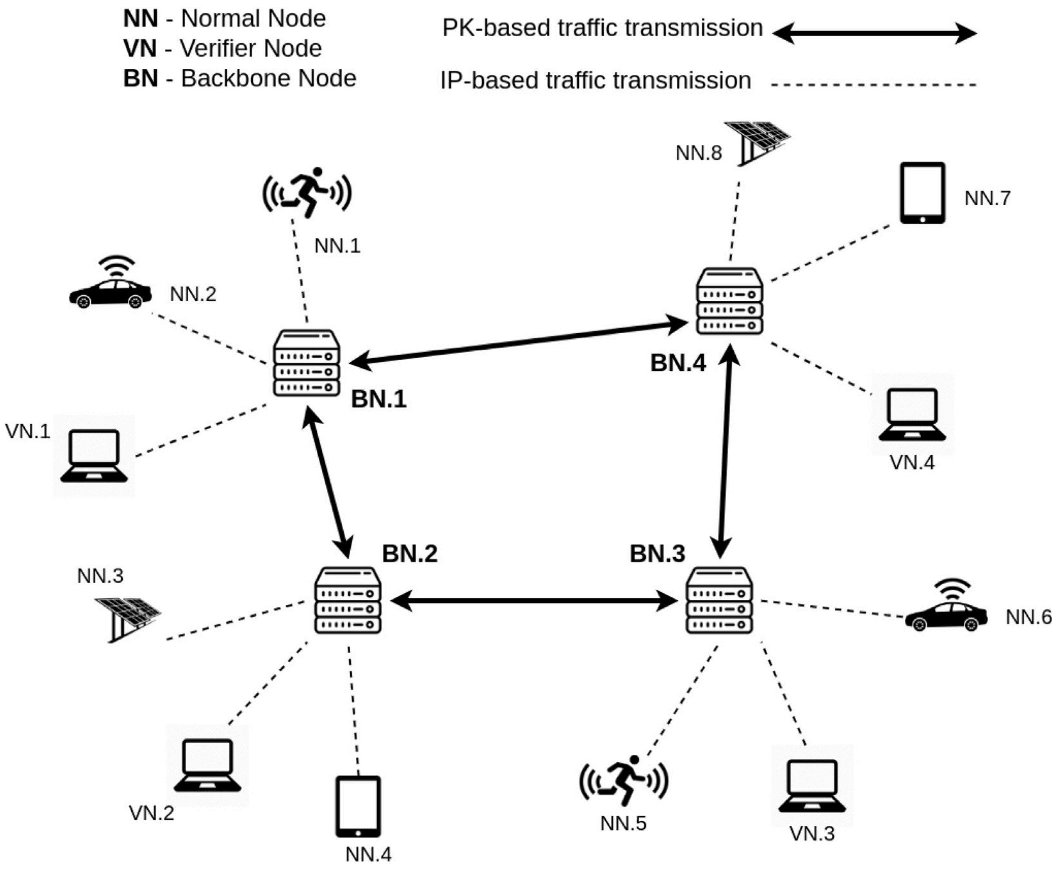


Fig. 5. A high level view of Vericom.

The trusted backbone nodes communicate to form the backbone network. We assume an authorized organization, e.g., the government or the Internet backbone network provider, connects the potential backbone nodes. Vericom routes traffic based on the hash function output of the traffic as outlined in the verification layer.

4.2.2. Untrusted backbone nodes

In this scenario, we assume that the backbone nodes are not trusted, neither to other backbone nodes nor to the network participants. This in turn makes security challenging as a malicious backbone node may attempt to drop incoming/outgoing traffic. Note that as blockchain transactions are sealed using asymmetric encryption, modification of the transaction/block content by the malicious backbone nodes is not possible. To enhance the security of Vericom in the presence of untrusted backbone nodes, we employ neighbor monitoring algorithms (such as in [28]) where the behavior of each backbone node is monitored by its neighbors which in turn can detect malicious activities. The neighbor nodes monitor the volume of incoming and outgoing traffic to a backbone node and identify the cases where the backbone node may drop all or selected packets. The monitoring algorithm presented in [28] does not require the analysis of the content of the packets in the network and the results in this work depict that power consumed is negligible. Thus, the computation burden on the neighbor nodes increases minimally due to monitoring. We introduce a random and unpredictable method to select neighbors for monitoring the behavior of each node to enhance the security as compared with conventional neighbor monitoring methods. To identify the backbone nodes and the corresponding monitoring nodes, Vericom relies on hash function output as outlined later in Section 4.3. In summary, a smart contract calculates a weight value corresponding to the PK of each node and orders the interested nodes in a descending list. The first node in the list is selected as the backbone node and the following *n* nodes in the list are selected as monitoring nodes for the first node (assuming that *n* is the number of nodes that shall monitor a backbone node). This process continues to cover all interested nodes. The neighbors will also be incentivized to participate in backbone network by receiving a fee (as discussed later in this section).

The backbone nodes receive traffic from the source IoT nodes and route them to the proper destination, that is identified based on the verification layer (see Algorithm 1). To receive traffic, the nodes shall join at least one backbone node. Otherwise, they will fail to receive traffic and thus will be isolated from the network.

Each validator or verifier orders the list of backbone nodes based on the delay experienced to reach them and sends a join request to the backbone node with minimum delay. The backbone nodes have limited resources and thus can serve limited number of nodes (depending on the amount of available resources and the volume of traffic in the network). To avoid queuing and thus reduce communication delay, each backbone node accepts join requests only from a particular number of nodes. Once the maximum number of nodes join, the backbone node rejects the join requests from the new nodes. The nodes then send the join request to the next backbone node in their list. This ensures the minimum delay in routing traffic in the network. Once all nodes join the backbone network, each backbone node broadcasts the PK of its corresponding nodes to the rest of the backbone nodes. The backbone nodes maintain a routing table that stores the list of nodes associated with other backbone nodes and their corresponding role (i.e., validator or verifier). During a particular time-interval, known as route update interval (RUI), the backbone nodes broadcast an update packet that contains the updated list of nodes connected to the backbone node. This in turn ensures that the backbone nodes can update the routing information in case of any change to the underlying nodes. To reduce packet overhead, each backbone node only generates an RUI if changes

Table 3
The routing table of BN.1 in Fig. 5.

Validator ID	Next hop
VN.4	4
VN.2	2
VN.3	2

happened to the underlying nodes list since the last RUI (or initialization). As an example the routing table of BN.1 in Fig. 5 is as shown in Table 3. To reduce the size of the routing table, each BN may only store the next hop in the path to reach a particular verifier (which is similar to routing tables in the Internet). The backbone nodes employ conventional routing algorithms such as OSFP [RFC 2328] to decide on the next hop node toward the destination.

4.2.3. Incentives

The backbone nodes dedicate communication and computational resources to manage the traffic flow that in turn incurs monetary costs. To incentivize nodes to join the backbone network, we introduce a *Traffic Management Fee (TMF)*. TMF is paid by the validators based on the total number of blocks they generated during an epoch time known as Θ. In our setting, $\Theta = consensus\_period$ (see Section 3). At the end of a Θ each validator calculates TMF as:

$$TMF = TF * ledger\_length \tag{1}$$

where *Traffic fee (TF)* is the cost for forwarding the traffic related to a single block and is defined by the blockchain designers. The size of the block and the bandwidth cost are two key factors that impact TF value. *Ledger_length* is the total number of the blocks each validator has generated during Θ. Recall that we base Vericom on tree-chain where each validator stores blocks in its own ledger. The validator then pays TMF to a *Traffic Accounting (TA)* smart contract. TA is hard coded by the blockchain designers in the first block in the blockchain that is known as the *genesis block*.

TA collects TMF from the validators and then equally distributes between the backbone nodes. Upon receipt of the payment from the validator, TA verifies if the validator has paid the right amount by calculating the TMF and matching with the received fund. If a validator pays less or no fee to the TA contract, the TA will notify the rest of the network by broadcasting a transaction. The validator will be denied from joining the validators in the next consensus period round until payment is made.

As evident from the above discussion, the only penalty for a validator that pays no/less fee is to prevent them from functioning as validator in future. Note that TMF is small and the benefits of being a validator are much higher than the TMF (as the validators collect transaction fees). An alternative option is to block specific amount of money from each validator initially and deduct from the blocked money in case the validator fails to pay the fee.

As outlined above, the transmission layer receives the traffic from the source nodes and multicast to dedicated nodes that are identified by the *verification layer*.

4.3. Verification layer

The verification layer aims to identify the destination of the traffic received by the backbone nodes. This reduces the computational overhead associated with transactions/block verification, compared to conventional blockchains, where all nodes must verify the new transactions or blocks. The verification of new transactions or blocks typically involves matching the signature with the PK of each transaction. To protect against malicious nodes that may falsely claim a fake transaction/block as valid, the destination nodes are selected randomly, dynamically, and uniquely for each transaction/block. The destination is either (i) validator set that verifies and commits new transactions in the blockchain by following the consensus algorithm, or (ii) verifier set that is a group of nodes that verify newly generated blocks. Both these sets are randomly and dynamically selected from the same pool of nodes in the network based on the hash of the traffic as outlined below.

Table 4
An example of WD.

α	$\omega(\alpha)$
a–z	0–25
A–Z	26–51
0–9	52–61

To identify the validator/verifier set corresponding to each transaction/block, Vericom relies on the hash function output of the traffic. Each potential character in the hash function output is allocated a particular weight that is identified in a *Weight Dictionary (WD)* an example of which is shown in Table 4. During the bootstrapping, the participating nodes in the blockchain (referred to as *PN*) that are interested to function as validator/verifier broadcast a *validator interest transaction* that contains their PK. The nodes shall broadcast the interest transaction within a particular time frame referred to as γ, defined by the network designers. To ensure consistency among the participating nodes, during the network setup, a *Validation Range Distributor (VRD)* smart contract is deployed in the genesis block, i.e., the first block in the blockchain. *Validation Range (VR)* is a range of *Most Significant Character (MSCh)* of the hash function output. To communicate with VRD, the participating nodes shall populate the address of the VRD, i.e., the hash of its content, as the destination in their transaction. The interested nodes send their PK to the VRD.

At the end of γ, the VRD smart contract starts calculating a Key Weight Metric (KWM) corresponding to the each received PK. KWM is calculated as follows (k is the size of the hash function output):

$$\sum_{i=1}^{k} fw(\alpha_{\mathrm{i}}) \tag{2}$$

where $fw(\alpha)$ is the final weight of α which is calculated as:

$$fw(\alpha) = \begin{cases} \omega(\alpha) & \text{if } r = 0 \\ \omega(\alpha) * (0.2^{r}) & \text{if } r > 0 \end{cases} \tag{3}$$

where r is the number of times that α is repeated in the hash function output. This in turn ensures that the final KWM corresponding to each PK is unique. The VRD then creates a descending list of the PKs based on the KWM values and allocate a particular VR. The size of VR is defined as follows:

$$\lceil \frac{number\_of\_validators}{62} \rceil$$

where 62 is the total number of possible values in each byte of the hash output which is from the following range: {0,..,9,a, … ,z,A, … ,Z}. The above division may have fractions. In that case, the first node in the list will be allocated a larger VR to accommodate all values. As an example, in a network with 10 nodes, the size of the VR allocated to the first node is 8 and the size of VR for other nodes is 6. The VR is allocated to the validators based on the position of PK of the validator in the KWM list, e.g., the validator with the highest value of KWM is allocated to the first VR. The validators form a Distributed Hash Table (DHT) that includes the PK of each validator and its corresponding VR that is used to identify the corresponding VR to each validator during for routing traffic.

When a backbone node receives traffic from the source node, it decides on the verifier/validator set by evaluating the output of the hash of the traffic, i.e., block or transaction content. Each set contains a main node, which acts as the leader of the nodes in the set, and the sub nodes which are the nodes that monitor the behavior of the main node. The backbone node first must identify the main node in the set.

For transaction t_i, the main validator is the validator whose VR covers the MSCh of $h(t_i)$ where $h(x)$ represents the hash function output of x (Step 2 Fig. 4). Once the main validator is identified, the backbone nodes add n successors and predecessors of the main validator to the validator set as sub validators defined in the DHT corresponding to the VR allocation. $1 \leq n \leq N/4$ where N refers to the total number of validators in the network. The upper bound value for n is identified in a way to ensure there will be no overlap between validator set and verifier set (see discussion in the rest of this section). Successors and predecessors are defined as the nodes that are immediately after and before the main validator in DHT according to the KWM. The value of n depends on the application. Larger n increases security to colluding malicious nodes in the validator sets, but also increases computational overhead for verifying and committing new transactions. The validator set first has to verify the transaction that involves matching the PK of the transaction with the corresponding signature (step 3, Fig. 4). Depending on the application, other steps might also be involved, e.g. if a transaction is chained to a previous transaction, the verifier shall verify the existence of the previous transaction. After verification, the main validator commits the transaction in the blockchain into a new block (following Tree-chain algorithm). The sub validators monitor the behavior of the main validator and inform the rest of the network in case any malicious activity, e.g., not storing new transactions or storing fake transactions, is detected. In case a different consensus algorithm other than Tree-chain is employed, the validators will add the transactions that are already verified by the verifier set to the pool of pending transactions.

Once the new block is committed, the validator sends the block to its corresponding backbone node to be verified (step 4, Fig. 4). The block must be signed by the nodes in the verifier set to be considered as a valid block in the blockchain. Fig. 6 shows the steps involved in verifying new blocks. Similar to validator set, the verifier set is identified based on the hash of the block. The verifier set consists of the main verifier and subverifiers which are m successors and predecessors of the main verifier. To enhance the security of Vericom against malicious nodes that may attempt to commit fake blocks and verify them as true blocks, the underlying nodes in the validator set shall always be different from the verifier set. In case the validator and verifier set have overlapping nodes, the main verifier will be:

$$MainVerifier = \begin{cases} 2n & \text{if } n > m \\ 2n + m & \text{if } n \leq m \end{cases} \tag{4}$$

This ensures that there will be no overlapping nodes in two sets. Once the nodes in the verifier set received the new block ((step 5, Fig. 4)), they first verify the block and then sign and send it to the main verifier (see Fig. 6). The verification of the block involves verifying the underlying transactions and the block header (step 6, Fig. 4). The transaction verification is as outlined earlier. The block header verification involves matching the PK of the block generator with the corresponding signature. Next, the verifier checks if the hash of the block content falls within VR of the block generator. The main verifier adds the signatures of all the nodes in the verifier set to the block and multicast to the other validators in the network to be stored in the blockchain (step 7, Figs. 4 and 6). The validators may accept the signed block by the verifier set without verifying the underlying transactions. All nodes in the blockchain still can verify the traffic confirmed by the verifier set which in turn will detect any malicious activity (see Section 5).

In summary, the verification layer ensures that each transaction/ block is verified twice by randomly selected nodes: first by the transaction validator set and second by the block verifier set.

5. Evaluation and discussion

In this section we provide qualitative security analysis as well as quantitative performance evaluation.

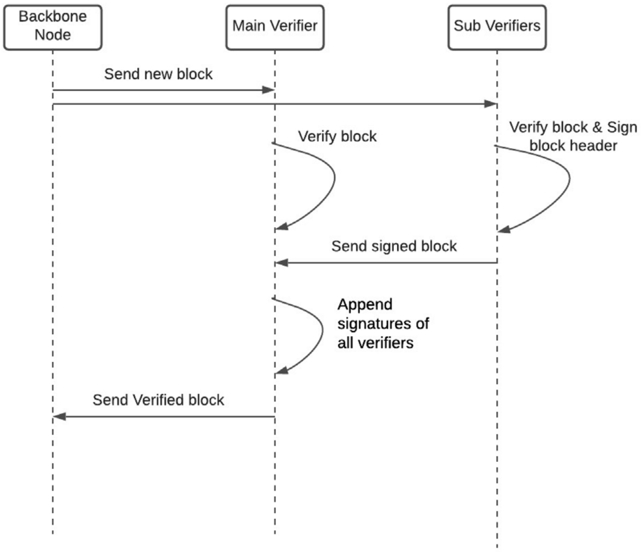


Fig. 6. The process of verifying new blocks.

5.1. Security and privacy analysis

Threat model: The malicious node can be any node in the network. In case of trusted backbone nodes, it is assumed that the backbone nodes function honestly and employ security safeguards that protects them against being compromised. The malicious node may drop the received traffic, generate fake transactions or blocks, and falsely claim that a transaction is valid. The malicious node may collaborate with other nodes to increase the chance of a successful attack.

Privacy: Vericom does not impact the anonymity level of the transactions stored in the blockchain. However, in conventional blockchains the transactions are broadcast thus the underlying nodes may not record the IP address of the underlying IoT nodes. In Vericom, the participating nodes shall record the IP of the backbone nodes which may potentially reveal information about the actual location of the backbone nodes. Note that the backbone nodes only forward traffic. If a backbone node wishes to generate a transaction, it employs a PK similar to other nodes which protects the anonymity of the node.

Recall that the backbone nodes multicast traffic to the intended group of recipients. This may potentially require the backbone node to record the IP address of the nodes in the verifier set. To prevent the backbone nodes from tracking the location based on IP, the verifiers may hide their IP using TOR browser [29].

Security: We discuss three security attacks:

False verification attack: In this attack, an IoT node or a group of nodes in the verifier set collaborate to mark a fake transaction or block as valid (step 3 & 6, Fig. 4). Recall from Section 4 that Vericom limits the number of nodes that verify a transaction or block to a verifier set. If a single verifier attempts to make a false claim, other verifiers will detect the malicious activity. The verifiers then broadcast a transaction to inform the rest of the network of the malicious behavior. The fake transaction is also broadcast to the network that enables the rest of the IoT nodes to verify the claim of the verifier set.

It may be possible that all the nodes in the verifier set collaboratively attempt to store fake transactions in the blockchain. As outlined in Section 4, once the main verifier commits the transaction in the blockchain, the block is sent to a new set of randomly selected nodes for verification which will detect the fake transaction (lines 9–11, Algorithm 1). If the new selected verifier set are malicious, they accept the block and mark it as valid. Vericom is designed for IoT that comprises millions of nodes, thus, it is expected that a large number of nodes will participate in the blockchain management which in turn makes it challenging for the malicious nodes to compromise all nodes in a randomly selected verifier set. Recall that any node may attempt to verify the transactions and blocks that are already verified by the verifier set. Such nodes will detect the malicious behavior. Given the large scale of IoT, it is highly likely that at least one node will allocate resources to verify transactions, this can be the SPs to ensure secure and safe services.

Fake transaction storage attack: In this attack, the malicious node attempts to store a fake transaction in the blockchain. The verifier of each transaction is identified based on the hash of the transaction content which is random and unpredictable. The malicious node might have control over (or collaborate with) other verifiers. In such case, if the transaction hash falls within the consensus code range managed by the malicious nodes, they can mark the fake transaction as valid. The validators attempt to store the verified transactions in the blockchain.

New blocks are verified by a randomly selected verifier set which will detect the fake transaction. In the worst case, if the nodes in the verifier set collaborate with the malicious node, the fake transaction will be broadcast as a valid transaction. Note that other nodes in the network can still verify the transactions and may detect the fake transaction. If so, the transaction ID is broadcast to the network the malicious nodes, including the verifiers, will be removed from the verification layer. It also worth noting that Vericom is designed for IoT where the number of verifiers is expected to be large. Plus, the verifiers are always selected randomly in an unpredictable manner, thus, it is hard for the malicious nodes to control the block generator and the verifier set.

Dropping attack: In this attack, a backbone node drops the incoming traffic to prevent blockchain participants from receiving service (applicable only in untrusted backbone node's scenario). Recall from Section 4 that in the untrusted backbone node scenario, a group of nodes collaboratively manage the traffic where the main node forwards the traffic and other nodes monitor the behavior of the main node which in turn enable them to detect any malicious activity. In case of a collaborative attack, the IoT nodes will not receive any service, thus will inform the rest of the network. The backbone nodes then will reconstruct the backbone network and prevent the malicious nodes from joining the network.

5.2. Performance evaluation

In this section, we study the performance of Vericom. We implemented Vericom using NS3 [30] incorporated with crypto++ security library to implement security features. During the evaluation, a total of 1000 transactions have been generated by the participating nodes in the blockchain. We set the size of the verifier set as 3 in our experiments. Note that Vericom employs tree-chain [25] as the underlying consensus algorithm. Both tree-chain and Vericom incorporate fundamental changes to the conventional blockchains, thus we were unable to use the existing blockchain simulation platforms, such as Hyperledger [31] or Ethereum [32]. We compare Vericom performance with a *baseline* scenario which is similar to conventional blockchains where all transactions and blocks are broadcast and verified by all participating IoT nodes. As our focus is on maintaining distribution for IoT security, we refrain from comparing against hierarchical, clustered, or sharded blockchains, as they intrinsically recentralize security and trust. We studied the following metrics:

- Packet overhead: This is the cumulative packet overhead incurred on the participating IoT nodes and is measured by summing the size of the received traffic by these nodes.
- Verification processing time: This is the processing time taken from a verifier to verify the transactions and blocks in the blockchain. We disregard the processing time associated with other blockchain-related tasks, e.g., consensus, as Vericom is not impacted by those.
- Delay: This is the time taken for sending a transaction from IoT node *A* to IoT node *B*.
- Vericom overheads: This is the extra overheads incurred by Vericom and includes: (i) the increased block size to include the verifier set PK and signature, and (ii) the size of the routing table.
- Verifier set size: This metric evaluates the trade-offs between security and overheads involved in the number of nodes that participate in the verifier set.

Table 5
Evaluating the scaling performance.

Metric	Vericom	Baseline
Packet overhead	O(H)	O(N^2)
Verification processing time	O(V)	O(N)
Delay	O(H)	O(N)

H: Number of hops in the backbone network.
N: Number of IoT nodes in the blockchain.
V: Size of the verifier set.

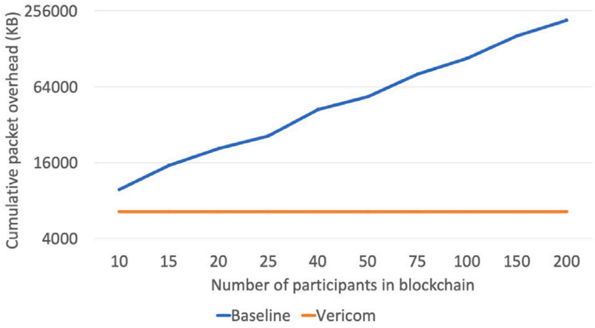


Fig. 7. The cumulative packet overhead.

We first evaluate the scaling performance metrics as a function of the key network parameters. Table 5 outlines the results. In Vericom, the packet overhead and delay in reaching another node depends on the number of hops in the backbone network while in baseline such overheads depend on the number of IoT nodes in the network which is significantly greater than Vericom. Similarly, the verification processing time in Vericom relies only on the number of nodes in the verifier set while the baseline is impacted by the number of participating IoT nodes in the network. In Vericom the packet overhead scales linearly with the number of hopes compared to quadratic overhead for conventional blockchain networks. In the worst case scenario where the nodes are connected linearly, i.e., in a chain structure, the packet overhead in Vericom will be *O(N)* as the packets should travel through all nodes.

Having discussed the scaling performance evaluation, we next evaluate the performance of Vericom based on the simulation output.

Packet overhead: The simulation results to evaluate the packet overhead are outlined in Fig. 7. We increase the number of underlying IoT nodes from 10 to 200 while the backbone network remains static as 20 nodes. As evident from the results, Vericom packet overhead remains constant as the number of IoT nodes increases at around 6000 KB, while the baseline packet overhead significantly increases reaching from 10,000 KB to around 216,000 KB. The main reason is that in Vericom the packets travel only among the backbone network and the verifier set and thus the increased in the number of underlying IoT nodes does not impact the general packet overhead, resulting in a reduction of packet overhead of up to nearly two orders of magnitude in this scenario over the baseline approach. However, in the baseline, the packets must be broadcast to the whole network and thus, as shown in Table 5, the packet overhead increases as the number of participating IoT nodes increases.

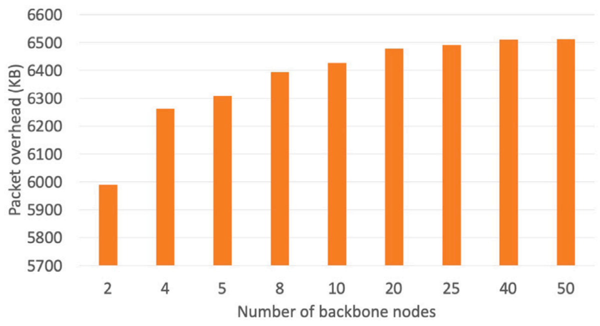


Fig. 8. Studying the impact of backbone nodes on packet overhead.

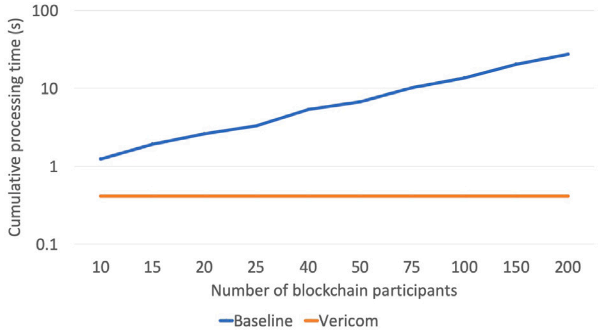


Fig. 9. Studying the cumulative processing time.

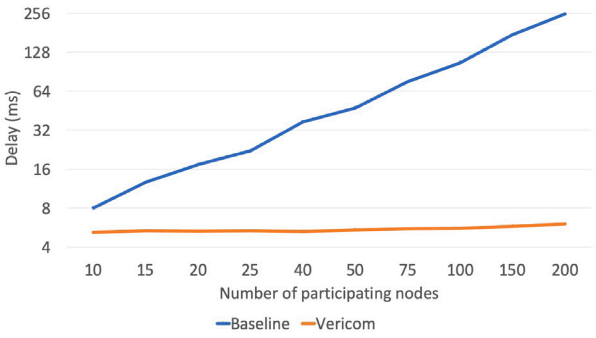


Fig. 10. Studying the communication delay.

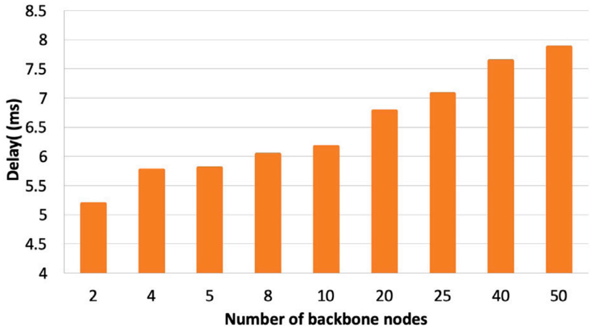


Fig. 11. Studying the communication delay while varying the number of backbone nodes.

Vericom packet overhead is largely impacted by the number of backbone nodes. Next, we change the number of backbone nodes in the network while the number of the underlying IoT nodes remains constant as 200 nodes. The new backbone nodes connect to a randomly selected existing backbone node. The source and the destination of the blockchain traffic are selected randomly that ensures the structure of the backbone network changes dynamically. The simulation results are shown in Fig. 8. The packet overhead increases from around 5900 KB with 2 backbone node to 6500 KB with 50 backbone nodes. Recall from Table 5 that Vericom packet overhead depends on the number of hops that the packets travel to reach a destination, thus the increased packet overhead for larger number of backbone nodes is relatively small.

Verification processing time: Fig. 9 compares the processing time in the baseline and Vericom to verify new transactions/blocks. Despite the fact that the underlying verifiers dynamically change, the number of nodes that verify new transactions/blocks, i.e., the size of the verifier set, always remains constant, that is 3 in our experiment, which results in a constant processing time in Vericom as the number of IoT nodes increases. However, increasing the number of IoT nodes in the baseline, increases the processing time as all the nodes verify new transactions/blocks.

Delay: The simulation results to evaluate the delay in communications between two nodes are represented in Fig. 10. We randomly select two nodes in the network to measure the delay while increasing the number of blockchain participants from 10–200 nodes. The delay in reaching another node in Vericom is in the range of [5–6] milliseconds while delay in the baseline is in the range of [8–253] milliseconds. Recall from Section 4 that Vericom routes packets among the backbone nodes to reduce the delay while in baseline, the packets are broadcast till reaching the destination which in turn significantly increases the delay in communications.

Recall from Table 5 that the delay in Vericom largely depends on the number of hops in a communication, thus we next study the impact of varying the number of backbone nodes on the communication delay. To prevent the traffic from traveling the same route while the number of backbone nodes increases, each new backbone node joins a randomly chosen backbone node and the source and destination of the traffic are randomly selected. Fig. 11 outlines the simulation results. A 25-fold increase in the number of backbone nodes from 2 to 50 nodes results in a 50% increase in communication delay.

Vericom overhead: Next we study the overheads incurred by Vericom. We first study the size of the routing table. As evident from the simulation results shown in Fig. 12, the size of the routing table increases from 0.9 KB to 8 KB while increasing the number of blockchain validators from 10 to 200. Recall from Section 4 that the routing table includes the PK of the validators and the IP address of the next hop backbone node for routing traffic. We next study the increased blocksize in Vericom to include the signature and PK of the verifiers. The size of the PK and signature in a transaction in our simulation is 459B. Thus, generally the incurred overhead can be measured as (459 $*$ V)B, where V is the size of the verifier set.

Verifier Set Size: Finally we study the impact of the size of the verifier set on the security of Vericom. We used NS3 to simulate a network where 50 nodes participate as verifiers. We run the simulation 5 times and the results are the average values. For the packet overhead we provided the error bar to show the changes in different runs. The upper and lower bound of the error bar depict the deviation of the maximum and minimum value of the packet overhead from the average value of the packet overhead. The probability of the attack remains constant in all runs. Recall that the verifier set are the selection of the nodes that verify the newly generated blocks. If even a single node in the verifier set detects a fake transaction, it will not sign the transaction which will prevent the block from being stored in the blockchain. Thus, for a successful attack, all nodes in the verifier set must collaborate. To reduce the chance of the attack, one may consider large verifier set, however, that in turn increases the processing overhead for verifying

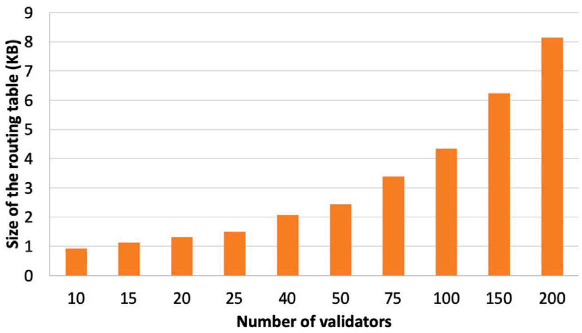
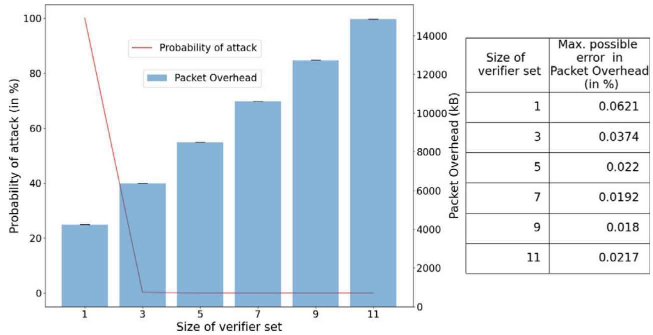


Fig. 12. Studying the size of the routing table in backbone nodes.



Size of verifier set	Max. possible error in Packet Overhead (in %)
1	0.0621
3	0.0374
5	0.022
7	0.0192
9	0.018
11	0.0217

Fig. 13. Studying the impact of the verifier set in Veriom security.

new blocks and the packet overhead in the network as the architecture will move toward broadcasting instead of multicasting. Fig. 13 outlines the impact of the verifier set on the packet overhead and the probability of a successful attack, i.e., the probability that all nodes in the verifier set are collaborative malicious nodes. As evident from the results, by increasing the size of the verifier set the probability of the successful attack decreases. Recall that the participating nodes in the verifier set are identified based on the hash function output which is completely random and unpredictable. The table in Fig. 13 depicts the magnitude of maximum observed error as a percentage of the average value of the packet overhead (in each verifier set size) over the 5 simulations. As the values in the table and Fig. 13 suggest, the error in the measurement of the packet overhead is small as reflected in Fig. 13. This depicts that the simulations run on the simulator do not vary greatly when repeated and therefore, the results obtained are precise.

To complement our simulation analysis, we now derive the probability of a successful attack in Vericom as a function of the network parameters. Consider a network with n verifiers where the size of the verifier set is r. In general the probability of a successful attack is calculated as follows:

$$P_r^n = \frac{C_1^n}{C_r^n}$$

where,

$$C_1^n = \frac{(n)!}{(n-r)!}$$

where,

$$(n)! = \prod_{i=1}^{n} i$$

The outlined formula can be employed to calculate the chance of successful attacks for a given network configuration and thus, decide on proper values for verifier set size.

6. Conclusion

In this paper we introduced a Verification and Communication architecture for IoT-based blockchain known as Vericom. Vericom introduces a distributed yet scalable blockchain architecture by introducing a PK-based traffic, i.e., transactions and blocks, multicasting algorithm which in turn reduces the bandwidth consumption of the underlying IoT nodes as compared with conventional blockchains where traffic is broadcasted to all nodes. Vericom incorporates two layers which are: (i) transmission layer where a backbone network is introduces to route traffic, and (ii) verification layer where traffic is verified by a randomly selected set of nodes that are unique for each transaction or block which in turn reduces the computational overhead associated with verifying new blocks and transactions. The simulation results show that Vericom reduces the blockchain packet and computational overhead by 90% in a network with 200 nodes which in turn facilitates the adoption of blockchain for low resource available IoT devices.

Interesting directions for future work on Vericom include better support for direct node-to-node communication and optimizing the blockchain storage. Vericom requires nodes to connect to a backbone node that manages their traffic. In applications that require direct communications between two nodes this may be challenging as the nodes require to know the hash of the transaction prior to receiving the same. Vericom considers optimizing traffic up to the point where blocks are verified. After that the blocks are broadcast. Comprehensive research is required to optimize the following steps that also involve blockchain storage. We expect Vericom will enable significant improvements in communication efficiency for blockchains.

Declaration of competing interest

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

References

[1] Z.-K. Zhang, M.C.Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, S. Shieh, IoT security: ongoing challenges and research opportunities, in: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, IEEE, 2014, pp. 230–234.

[2] M.A. Khan, K. Salah, IoT security: Review, blockchain solutions, and open challenges, Future Gener. Comput. Syst. 82 (2018) 395–411.

[3] W.H. Hassan, et al., Current research on Internet of Things (IoT) security: A survey, Comput. Netw. 148 (2019) 283–294.

[4] F.A. Alaba, M. Othman, I.A.T. Hashem, F. Alotaibi, Internet of Things security: A survey, J. Netw. Comput. Appl. 88 (2017) 10–28.

[5] O. Bouachir, M. Aloqaily, L. Tseng, A. Boukerche, Blockchain and fog computing for cyberphysical systems: The case of smart industry, Computer 53 (9) (2020) 36–45.

[6] H.-N. Dai, Z. Zheng, Y. Zhang, Blockchain for Internet of Things: A survey, IEEE Internet Things J. 6 (5) (2019) 8076–8094.

[7] D. Berdik, S. Otoum, N. Schmidt, D. Porter, Y. Jararweh, A survey on blockchain for information systems management and security, Inf. Process. Manage. 58 (1) (2021) 102397.

[8] S. Otoum, I. Al Ridhawi, H. Mouftah, Securing critical IoT infrastructures with blockchain-supported federated learning, IEEE Internet Things J. (2021).

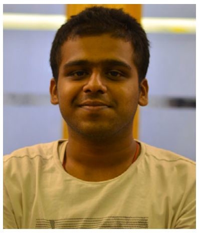
[9] K.-P. Yu, L. Tan, M. Aloqaily, H. Yang, Y. Jararweh, Blockchain-enhanced data sharing with traceable and direct revocation in IIoT, IEEE Trans. Ind. Inf. (2021).

[10] S. Nakamoto, Bitcoin: A Peer-To-Peer Electronic Cash System, Tech. rep., Manubot, 2019.

[11] M. Ma, G. Shi, F. Li, Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the IoT scenario, IEEE Access 7 (2019) 34045–34059.

[12] W. Tong, X. Dong, Y. Shen, X. Jiang, A hierarchical sharding protocol for multi-domain iot blockchains, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.
[13] A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, LSB: A lightweight scalable blockchain for IoT security and anonymity, J. Parallel Distrib. Comput. 134 (2019) 180–197.
[14] L. Tseng, L. Wong, S. Otoum, M. Aloqaily, J.B. Othman, Blockchain for managing heterogeneous Internet of Things: A perspective architecture, IEEE Netw. 34 (1) (2020) 16–23.
[15] J. Yun, Y. Goh, J.-M. Chung, DQN based optimization framework for secure sharded blockchain systems, IEEE Internet Things J. (2020).
[16] A. Dorri, C. Roulin, R. Jurdak, S.S. Kanhere, On the activity privacy of blockchain for IoT, in: 2019 IEEE 44th Conference on Local Computer Networks (LCN), IEEE, 2019, pp. 258–261.
[17] M. Khorasany, A. Dorri, R. Razzaghi, R. Jurdak, Lightweight blockchain framework for location-aware peer-to-peer energy trading, Int. J. Electr. Power Energy Syst. 127 (2021) 106610.
[18] Y.E. Oktian, S.-G. Lee, H.J. Lee, Hierarchical multi-blockchain architecture for scalable Internet of Things environment, Electronics 9 (6) (2020) 1050.
[19] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains, 2014, URL: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains72.
[20] A. Singh, K. Click, R.M. Parizi, Q. Zhang, A. Dehghantanha, K.-K.R. Choo, Sidechain technologies in blockchain networks: An examination and state-of-the-art review, J. Netw. Comput. Appl. 149 (2020) 102471.
[21] F.S. Ali, M. Aloqaily, O. Ozkasap, O. Bouachir, Blockchain-assisted decentralized virtual prosumer grouping for P2P energy trading, in: 2020 IEEE 21st International Symposium on" a World of Wireless, Mobile and Multimedia Networks"(WoWMoM), IEEE, 2020, pp. 385–390.
[22] S. Popov, The tangle, White Pap. 1 (2018) 3.
[23] W.F. Silvano, R. Marcelino, Iota tangle: A cryptocurrency to communicate Internet-of-Things data, Future Gener. Comput. Syst. 112 (2020) 307–319.
[24] X. Wang, X. Zha, W. Ni, R.P. Liu, Y.J. Guo, X. Niu, K. Zheng, Survey on blockchain for Internet of Things, Comput. Commun. 136 (2019) 10–29.
[25] A. Dorri, R. Jurdak, Tree-chain: a fast lightweight consensus algorithm for iot applications, in: 2020 IEEE 45th Conference on Local Computer Networks (LCN), IEEE, 2020, pp. 369–372.
[26] A. Dorri, R. Jurdak, Tree-chain: a lightweight consensus algorithm for iot-based blockchains, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2021, pp. 1–9.
[27] A. Dorri, F. Luo, S.S. Kanhere, R. Jurdak, Z.Y. Dong, SPB: A secure private blockchain-based solution for distributed energy trading, IEEE Commun. Mag. 57 (7) (2019) 120–126.
[28] T.H. Hai, E.-N. Huh, Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge, in: 2008 Seventh IEEE International Symposium on Network Computing and Applications, IEEE, 2008, pp. 325–331.
[29] TOR project. http://torproject.org/.
[30] Network Simulator (NS3). https://www.nsnam.org/.
[31] Ethereum. https://ethereum.org/en.
[32] Hyperledger Fabric. https://www.hyperledger.org/use/fabric.

Ali Dorri is a Research Fellow at Queensland University of Technology (QUT), Brisbane, Australia. He received his Ph.D. degree from the University of New South Wales (UNSW), Sydney, Australia. He was also a Postgraduate research student at CSIRO, Australia. His core publications on blockchain for IoT have received tremendous attention and one of his papers is continuously ranked among the most downloaded conference papers in IEEE explorer (top 50 and in some months the second rank). Two of his papers are top-cited in their respective venues. His publications are cited over 3300 times and Ali has h-index of 18. His research interest includes blockchain, Internet of Things (IoT), security and privacy, and distributed system. He has published over 31 peer-reviewed papers. Ali served on the organizing committee of SDLT and BCCA and as technical program committee in 10 conferences including ICBC.



Shailesh Mishra is an incoming Ph.D. candidate at the School of Computer and Communication Sciences, École polytechnique fédérale de Lausanne. He has recently completed his dual-degree (B.Tech+M.Tech) major in the Department of Electrical Engineering at the Indian Institute of Technology, Kharagpur. His current research interests include distributed systems, blockchain and its applications in IoT, security and data privacy. He is also working towards making blockchain more usable and scalable.

Raja Jurdak is a Professor of Distributed Systems and Chair in Applied Data Sciences at Queensland University of Technology, and Director of the Trusted Networks Lab. He received the Ph.D. in information and computer science from the University of California, Irvine. He previously established and led the Distributed Sensing Systems Group at CSIRO's Data61, where he maintains a visiting scientist role. His research interests include trust, mobility and energy-efficiency in networks. Prof. Jurdak has published over 220 peer-reviewed articles and 2 authored books that have collectively been cited over 10,000 times, with an h-index of 45. His work on blockchain in IoT is among the most cited globally in this area. He was an Embark fellow in 2006, an Endeavour Fellow in 2011, and a Eureka prize finalist in 2019. Prof. Jurdak was TPC chair of IEEE ICBC 2021, serves on the editorial board of Ad Hoc Networks and Nature Scientific Reports, and has been a visiting academic at Oxford and MIT. He regularly serves on the organizing and technical program committees of top international conferences, including Percom, ICBC, IPSN, WoWMoM, and ICDCS. He is a conjoint professor with the University of New South Wales, and a senior member of the IEEE, and a Distinguished Visitor of the IEEE Computer Society.