Journal of Parallel and Distributed Computing 134 (2019) 180–197

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

LSB: A Lightweight Scalable Blockchain for IoT security and anonymity

Ali Dorri [a,*], Salil S. Kanhere [b], Raja Jurdak [c], Praveen Gauravaram [d]

[a] The School of computer science and engineering, UNSW, Sydney and DATA61 CSIRO, Australia
[b] The School of computer science and engineering, UNSW, Sydney, Australia
[c] School of Electrical Engineering and Computer Science, QUT and DATA61, CSIRO, Brisbane, Australia
[d] Tata Consultancy Services, Brisbane, Australia

ARTICLE INFO

Article history:
Received 22 October 2018
Received in revised form 1 July 2019
Accepted 17 August 2019
Available online 9 September 2019

Keywords:
Internet of Things
Blockchain
Security
Smart home

ABSTRACT

In recent years, Blockchain has attracted tremendous attention due to its salient features including auditability, immutability, security, and anonymity. Resulting from these salient features, blockchain has been applied in multiple non-monetary applications including the Internet of Things (IoT). However, blockchain is computationally expensive, has limited scalability and incurs significant bandwidth overheads and delays which are not suited for most IoT applications. In this paper, we propose a Lightweight Scalable blockchain (LSB) that is optimized for IoT requirements while also providing end-to-end security. Our blockchain instantiation achieves decentralization by forming an overlay network where high resource devices jointly manage the blockchain. The overlay is organized as distinct clusters to reduce overheads and the cluster heads are responsible for managing the public blockchain. We propose a Distributed Time-based Consensus algorithm (DTC) which reduces the mining processing overhead and delay. Distributed trust approach is employed by the cluster heads to progressively reduce the processing overhead for verifying new blocks. LSB incorporates a Distributed Throughput Management (DTM) algorithm which ensures that the blockchain throughput does not significantly deviate from the cumulative transaction load in the network. We explore our approach in a smart home setting as a representative example for broader IoT applications. Qualitative arguments demonstrate that our approach is resilient to several security attacks. Extensive simulations show that packet overhead and delay are decreased and blockchain scalability is increased compared to relevant baselines.

1. Introduction

Blockchain is an immutable, auditable, and timestamp ledger of blocks that is used for storing and sharing data in a distributed manner [30]. The stored data can be payment history, e.g. Bitcoin [38], or a contract [51] or even personal data [54]. In recent years, blockchain has attracted tremendous attention from practitioners and academics in different disciplines (including law, finance, and computer science) due to its salient features which include a distributed structure, auditability, immutability and security and anonymity [9].

Blockchain maintains a distributed digital ledger of transactions that is shared across all participating nodes. Storing all transactions in the ledger provides high auditability of the transactions. New transactions are verified and confirmed by other nodes participating in the network, thus eliminating the need for a central authority. Special nodes in the network, known as miners, add newly generated transactions to a pool of pending transactions. Each miner collates pending transactions into a block when the size of the collected transactions reached a predefined size known as block size. Appending a new block to the blockchain (referred to as mining in literature) entails following a consensus algorithm that ensures blockchain security against malicious miners and introduces randomness among them. Existing blockchain implementations typically use either: Proof of Work (POW) [50] or Proof of Stake (POS) [51]. POW demands high computational resources, thus the miner with higher computational power is more likely to win the consensus algorithm and mine the next block. POS demands the miners to lock their stake, or assets, in the blockchain to mine new blocks. Miners with a higher locked stake are more likely to mine the next block. Each user employs a Public Key (PK) as his identity while creating transactions that in turn introduces high-level of anonymity.

In recent years, blockchain has attracted tremendous attention to enhance security, auditability, reliability, and anonymity of the Internet of Things (IoT) where billions of everyday devices are connected to the Internet to facilitate everyday life and offer

* Corresponding author.
E-mail addresses: ali.dorri@unsw.edu.au (A. Dorri), salil.kanhere@unsw.edu.au (S.S. Kanhere), r.jurdak@qut.edu.au (R. Jurdak), p.gauravaram@tcs.com (P. Gauravaram).

https://doi.org/10.1016/j.jpdc.2019.08.005
0743-7315/

personalized services. Conventional IoT frameworks suffer from the following challenges:

- Resource consumption: Most IoT devices have limited resources, including bandwidth, computation, and memory, which is incompatible with the requirements of complex security solutions [56].
- Centralization: Current IoT ecosystems rely on centralized brokered communication models where all devices are identified, authenticated and connected through cloud servers. This model is unlikely to scale as billions of devices are connected. Moreover, cloud servers will remain a bottleneck and point of failure that can disrupt the entire network [56].

The various benefits afforded by blockchain technology as outlined earlier in this section make it an attractive solution for addressing the aforementioned problems in IoT. IBM introduced a new blockchain instantiation for IoT known as Hyperledger Fabric [14], which is a permissioned blockchain, wherein only authorized nodes can participate in blockchain. The authors in [28] proposed a new framework for managing the IoT devices using smart contracts. This framework manages each device by a smart contract. In [44], the authors proposed a blockchain-based Software Defined Network (SDN) architecture for IoT. The architecture manages access requests using rule tables which are stored in the blockchain.

However, the existing blockchain instantiations suffer from the following limitations:

Scalability and overheads: In a typical blockchain implementation, all blocks are broadcast to and verified by all nodes. This leads to significant scalability issues since the broadcast traffic and processing overheads would increase quadratically with the number of nodes in network. The associated overheads are intractable as many IoT devices have limited bandwidth connections (e.g. Low Power Wide Area Networks such as LoRa [37]) and processing capabilities.

Complex consensus algorithms: Most of the consensus algorithms employed in blockchain consume significant resources of the participating nodes which are far beyond the capabilities of most IoT devices.

Latency: There is a non-trivial delay associated with ensuring that a transaction is confirmed by nodes participating in the blockchain. For example, in Bitcoin, it can take up to 30 min for a transaction to be confirmed. Most IoT applications have stricter delay requirements e.g. a service provider requesting data from a smart home sensor should not have to wait for several minutes as the data are processed to offer real-time services to the user.

Throughput: In blockchain, the throughput is defined as the number of transactions that can be stored. Classical instantiations of blockchain have limited throughput. For example, Bitcoin throughput is 7 transactions per second. However, the number of transactions in the IoT ecosystem would far exceed such limits due to extensive interactions between various entities.

The main contribution of this paper is to address the outlined challenges by proposing a Lightweight Scalable Blockchain (LSB) for IoT. To ensure scalability, the overlay nodes, which include IoT devices, cloud storages, and Service Providers (SPs), are organized into clusters and only the Cluster Heads (CH) are responsible for managing the blockchain. Blockchain management involves verification and storage of individual transactions or blocks of transactions. A transaction is defined as the basic communication primitive for exchanging control information between nodes. Transaction are collated to form a block which is then appended to the blockchain to form the distributed ledger. In LSB, the data of IoT devices is stored off-the-chain, e.g., in cloud storage, to reduce the memory footprint and packet overhead of the blockchain. Additionally, the flow of data to and from the IoT devices is kept separate from the transaction flow. Transactions are broadcast among the overlay nodes while data packets are routed toward their destination. This separation allows optimal unicast routing of data packets, thus resulting in reduced delays and packet overhead for transferring data. We propose a Distributed Time-based Consensus (DTC) algorithm that requires each CH to wait for a random time before mining, i.e., appending, a block in place of solving a computationally demanding puzzle, and significantly reduces the mining processing overhead. DTC limits the number of new blocks generated by the CHs within a tunable consensus period to protect against malicious CHs which may flood the network with a large number of new blocks. To reduce the computation overhead associated with verifying new blocks that are to be added to the blockchain, LSB employs a distributed trust algorithm. Each CH accumulates evidence about other CHs based on the validity of new blocks that they generate. The number of transactions in a new block that need to be verified is gradually reduced as CHs develop trust in each other. Finally, we propose a Distributed Throughput Management (DTM) mechanism to dynamically adjust certain system parameters to ensure that the throughput of the public blockchain (i.e., the number of transactions appended to the blockchain) does not significantly deviate from the transaction load in the network. DTM ensures that network is self-scaling, i.e., as the network grows in size, more transactions can be appended to the public blockchain, thus increasing the throughput.

Table 1
A summary of acronyms used in the paper.

Acronym	The phrase
LSB	Lightweight Scalable Blockchain
IoT	Internet of Things
DTC	Distributed Time-based Consensus algorithm
DTM	Distributed Throughput Management algorithm
PoW	Proof of Work
PoS	Proof of Stake
PK	Public Key
SP	Service Provider
CH	Cluster Head
OBM	Overlay Block Manager
T_ID	Transaction identifier
P_T_ID	Previous transaction identifier

The key contributions of this paper are summarized below:

1. We present a new blockchain instantiation, known as Lightweight Scalable Blockchain (LSB), that is tailored to meet the specific requirements of IoT devices and applications. We incorporate a number of optimizations which include a distributed time-based consensus algorithm, a distributed trust method, a distributed throughput management strategy and separation of the transaction traffic from the data flow.
2. We undertake a qualitative analysis of LSB against 8 relevant cyber attacks and outline the specific defense mechanisms, which ensure that LSB is resilient against all of them. Additionally, a risk analysis is conducted to investigate the likelihood of the attacks.
3. We conduct extensive simulations using NS3 to evaluate key performance parameters including latency, processing time, and resilience against cyber attacks. Our results justify various design decisions made and demonstrate the efficacy of the proposed optimizations.

Table 1 proposes a summary of the frequently used acronyms in the rest of the paper.

The rest of the paper is organized as follows. Section 2 outlines details of LSB. Section 3 discusses a case study of LSB in a smart home. Detailed security analysis and performance evaluations are

presented in Section 4. Section 5 discusses further aspects of LSB. Section 6 presents a literature review on IoT security and blockchain applications, and finally Section 7 concludes the paper and outlines future work.

2. Lightweight Scalable Blockchain (LSB)

In this section, we discuss the LSB in detail. The basic communication primitive for exchanging control information among any entities is referred to as a transaction. We first discuss an overview of the LSB in Section 2.1, following transactions and blocks are outlined in Section 2.2. Next, Section 2.3 outlines time-based consensus algorithm. Verification of transactions and distributed trust is discussed in Section 2.4. Section 2.5 gives details of distributed throughput management. A high-level summary of multiple algorithms in LSB is given in Section 2.6. Finally, Section 2.7 outlines compatibility of LSB with IoT requirements.

2.1. Overview

The participants in blockchain, which includes IoT devices, SPs, and IoT users, jointly form an overlay network as shown in Fig. 1. Similar to Bitcoin, we assume that each node in the overlay is known by a Public Key (PK). Nodes use a fresh PK to generate each new transaction to ensure anonymity (discussed further in Section 4.1). As noted in Section 1, IoT devices have limited resources, thus, verifying all new blocks and transactions may be far beyond their capabilities. To ensure scalability and reduce processing and packet overhead on IoT devices, we assume that the blockchain is managed by a subset of the overlay nodes. We assume that a clustering algorithm such as in [31] is used to group nodes into clusters, with each cluster electing a Cluster Head (CH). In [31] the CHs will be the nodes with maximum number of one-hop neighbors so that the CH will have the maximum coverage in the network and the network can be clustered with the minimum number of CHs. In case that multiple nodes have the same number of neighbors, the cluster members randomly join one of the clusters. Eventually, the node with higher number of cluster members will be chosen as the CH. The cluster members also choose a co-leader for the cluster. The co-leader become the CH in case the previous CH departs from the network, for example, due to loss of connectivity. A node selected as a CH is expected to remain online for an extended duration of time and to have sufficient resources for processing blocks and transactions. CHs are responsible for managing the blockchain and are thus referred to as Overlay Block Managers (OBMs). Managing the blockchain involves generation, verification and storage of individual transactions and blocks of transactions. Recall that blockchain is inherently a trustless system where the participating nodes do not trust each other. Thus, the cluster members monitor the behavior of their OBM. In case the cluster members detect any misbehavior, e.g., a transaction is not forwarded to other OBMs, they choose a new OBM (see dropping attack in Section 4.1). We assume that mechanisms such as those used in [46] for managing CH failures are in place.

As an illustrative example, consider that a smart thermostat wishes to communicate with the SP server. A transaction recording this interaction is recorded in the blockchain. The thermostat subsequently directly sends temperature readings to the SP server which analyzes the measurements and may send commands for adjusting the temperature to the thermostat. As noted in Section 1, in LSB data flow is distinct from transactions. Unlike transactions which are broadcast, the data packets are unicast and can be routed along optimal paths through the overlay network (between OBMs) using routing protocols such as OSPF [1]. For data packets, the recipient's OBM ID must be populated in the transaction which enables the OBMs to route the packet (see 2.2). This reduces the packet overhead for data exchange. To ensure data integrity, the transaction corresponding to the exchanged data between overlay nodes contains the hash of the data signed by the transaction generator.

In LSB, all transactions are stored on-chain. The data of IoT devices is stored off-the-chain to reduce the packet overhead and memory requirement of the blockchain. Each OBM stores a trust table (see Section 2.4), trust rate for other OBMs, and the timestamp of the last block generated by other OBMs off-the-chain.

2.2. Transactions and blocks

Transactions generated by an overlay node are secured using asymmetric encryption, digital signatures and cryptographic hash functions (e.g. SHA256). In blockchain, transactions can be further classified into: (i) *single signature* transactions which only contain the signature of the transaction generator, and (ii) *multisig* transactions which require more than one signature to be considered as valid transactions. A multisig transaction might require m out of n signatures, where n and m are number of signatures and $n >= m$, to be considered as valid. The value of m and n is defined based on application. An example application of a multisig transaction is given in Section 3.

The structure of a transaction is as follows:

T_ID || *P_T_ID* || *PK* || *Sign* || *output* || *metadata*

T_ID is the transaction identifier which is essentially the hash of the transaction content. *P_T_ID* is a pointer to the previous transaction of the same overlay node. Thus, all (multisig or single signature) transactions created by an overlay node are chained together and can be audited. This is followed by the *PK* and *signature* of the transaction generator. The signed *T_ID* is used as the signature of the transaction. In case of multisig transaction, the PK and signature of the other $(n-1)$ participating nodes precede the output field. Note that only m signatures (out of n) are required for the transaction to be considered as valid. The output field contains the hash of the PK that the transaction generator will use for its next transaction. This is necessary to verify the subsequent transaction created by the same node, since the overlay nodes change the PK used for generating each new transaction. *Metadata* fields provide extra information that participating nodes in the transaction might require, e.g., if the transaction is generated to request data of an IoT device for a specific period of time, the time period can be specified in the metadata (see Section 3 for further examples). The aforementioned fields are required for a transaction to be considered valid. However, additional fields can be defined based on the application.

Each overlay node must first create a genesis transaction, which serves as the starting point for its ledger in the blockchain, using one of the following approaches:

- Certificate authorities: In this approach, the node relies on the widely deployed Public Key Infrastructure (PKI) [18] in the Internet. The overlay node contacts a trusted Certificate Authority (CA), which ratifies the node's PK by attaching a signed certificate. The node includes the certificate in the genesis transaction. To verify the transaction, an OBM verifies the certificate. It is assumed that the OBMs have access to a list of trusted CA root certificates for verification (similar to browsers and operating systems).
- Burn coin in Bitcoin: Alternatively, if the node does not wish to rely on PKI, then it can privately create a genesis transaction by burning Bitcoins [49]. The node creates a permanent transaction in the Bitcoin blockchain by destroying a specified amount of fraction of coin (that can be defined as a design choice), which is referred to as "burning coins".

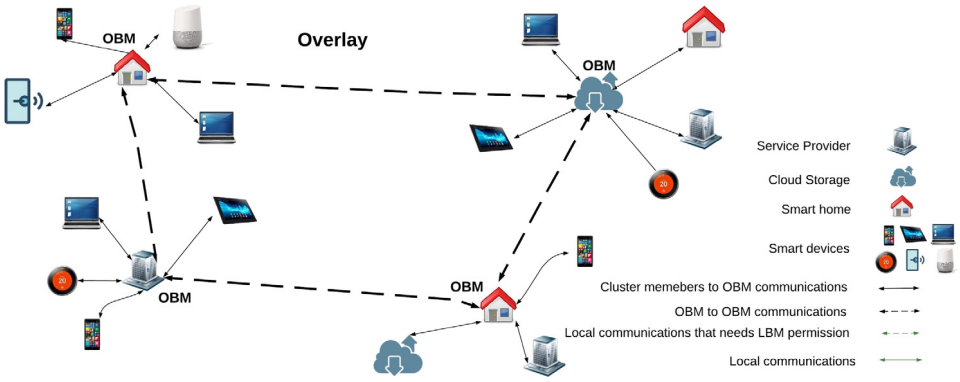

Fig. 1. An overview of LSB.

The address of the burn transaction is used as the input of the genesis transaction. The overlay node creates a genesis transaction with the same PK as the burn transaction and sends it to the OBM whose cluster it belongs to. If the genesis transaction generator is an OBM, then it broadcast the transaction to other OBMs. To verify the received genesis transaction, the OBM matches the PK of the genesis transaction with the PK of the burn transaction in the Bitcoin blockchain. Next, the OBM verifies the signature in the genesis transaction.

In both approaches after verification, the OBM broadcasts the genesis transaction to other OBMs to be stored in the blockchain. In LBS, the data flow is kept separate from the transaction flow.

Similar to Bitcoin, multiple transactions are grouped together and then processed as one block. A block can store at most *T_max* transactions. The value of *T_max* affects the blockchain throughput such that with a larger *T_max*, more transactions can be stored in a single block. Each block consists of two main parts namely, transactions and block header. The block header contains the following:

B_ID ‖ *P_B_ID* ‖ *B_Generator* ‖ *B_verifiers*

B_ID is the hash of the block content. *P_B_ID* is the hash of the previous block which ensures immutability. If an attacker attempts to change a previously stored transaction, then the hash of the corresponding block which is stored in the next block will no longer be consistent and will thus expose this attack. The rest of the fields will be discussed later in this section.

Recall that to ensure scalability, transactions and blocks are broadcast only to the OBMs. However, in case of multisig transactions, the transaction is not valid until m out of n participating nodes in the transaction have appended their signature. Thus, the transaction must be sent to the involved parties. To address this challenge, OBMs maintain a key list (essentially a simplified access control list) consisting of a list of PKs that can access the cluster members. If one of the PKs of a received transaction by the OBM matches with one of the keys in the list, the OBM broadcasts the transaction to its cluster members. The reason for broadcasting transactions in the cluster is to protect the privacy of the cluster member against malicious OBMs which may track the transactions received by a particular cluster member to link different transactions and thus deanonymize the node. If one of the PKs of the incoming transaction Y matches with an entry in the key list, then the OBM sends the transaction to the cluster member that has previously uploaded the key in keylist. Otherwise the transaction is broadcast to all other OBMs. All pending transactions are stored in a transaction pool at each OBM. When the size of the running pool becomes equal to *T_max* the OBM starts the process of creating a new block using a distributed time-based consensus (DTC) algorithm.

2.3. Distributed Time-based Consensus (DTC) algorithm

As noted in Section 1, in LSB, we propose a time-based consensus algorithm in place of the more resource-intensive alternatives such as PoW. The consensus algorithm must ensure a block generator is selected randomly among nodes and is limited in the number of blocks it can generate. This protects against malicious block generators which continuously generate new (fake) blocks. To introduce randomness among block generators, each OBM must wait for a random time, known as *waiting-period*, prior to generating a new block. To prevent OBMs from always claiming to have a short waiting-period, the neighboring OBMs monitor the frequency with which an OBM generates new blocks in the beginning of the waiting-period. If the number of such blocks exceeds a threshold, defined based on the application by the blockchain designers, the OBMs drop the block generated by their neighbor. Since the waiting-period differs for each OBM, an OBM might receive a new block created by another OBM that contains some or all of the transactions that are currently within the pool of transactions of the OBM. In this instance, this OBM must remove these transactions from its pool as they are stored in the blockchain by another OBM. Requiring OBMs to wait for a random time also reduces the number of duplicate blocks that can be generated simultaneously. The maximum waiting-time is capped at twice the maximum end-to-end delay in the overlay network. When a new block is generated, it is broadcast to all other overlay nodes so that it can be appended to the blockchain.

It is possible that multiple nodes generate a block in the same time and broadcast to the network which potentially would lead to blockchain forking. In the latter, the blockchain is diverged into two paths. To prevent forking, LSB relies on the concept of *the longest ledger* where the ledger that more blocks are chained to it is accepted as the true ledger in the blockchain. This concept is utilized in most of the existing blockchains to handle fork.

To protect the overlay against a malicious OBM that may potentially generate a large number of blocks with fake transactions which is referred to as an appending attack (discussed in Section 4.1), the periodicity with which an OBM can generate blocks is restricted such that only one block can be generated over an interval denoted by *consensus-period*. The consensus-period is adjusted by Distributed Throughput Management (DTM) and is discussed in Section 2.5. The default (and maximum) value for consensus-period is 10 min, which is similar to the mining interval in Bitcoin. The minimum value of consensus-period is equal to twice the maximum end-to-end delay in the overlay, to ensure that there is sufficient time for disseminating a new block generated by other OBMs. Each OBM monitors the frequency with which other OBMs generate blocks. Any non-compliant blocks are dropped and the trust associated with the responsible OBM is decreased as outlined in Section 2.4.

2.4. Verification

An OBM must validate each new block that it receives from other OBMs prior to appending it to the blockchain. To validate a block, the OBM first validates the signature of the block generator. It is assumed that each OBM uses a pre-defined key for generating blocks and it is assumed that these keys are known to all other OBMs [34]. Next, each individual transaction in the block is verified. A block is considered to be valid if all transactions contained in the block are valid.

Algorithm 1 outlines the procedure for verifying an individual transaction, X. The hash of the PK that will be used for the next transaction is stored in the output field of the current transaction. This ensures anonymity as the transaction generator can change his PK for each new transaction. Thus, the OBM first confirms this by comparing the hash of the PK in X with output of the previous transaction in the same ledger (lines 1, 2). Following this, the signature of X, that is contained within the fourth field of X, is verified (also called redeemed) using its PK in X (lines 4, 5). In case of multisig transaction, the signature of all participating nodes are verified using their PKs. If the steps complete successfully, X is verified.

Algorithm 1 Transaction verification.

Input: Overlay Transaction (X)
Output: True or False
Requester verification :
1: **if** (hash (X.PK) $\neq X_{-1}$.output) **then**
2: **return** False;
3: **else**
4: **if** (X.PK *redeem* X.signature) **then**
5: **return** true;
6: **end if**
7: **end if**

Verifying all transactions and blocks is computationally demanding, particularly when the number of nodes in the overlay increases. In the IoT context, one can expect serious scalability issues since the number of nodes is expected to be very large. Recall from Section 2 that LSB clusters the overlay network and only CHs, i.e., OBMs, manage the blockchain which in turn reduces packet overhead and improves scalability. However, the overhead associated with verifying each new block (which effectively entails verifying all transactions generated in the network) can be significant and would impact the scalability of the blockchain. To address this, LSB uses a *distributed trust algorithm* that gradually reduces the number of transactions that need to be verified in each new block as OBMs build up trust in one another. The algorithm introduces the notions of direct and indirect evidence as follows:

Direct evidence: OBM *A* has direct evidence about OBM *B* if it previously verified at least one block that was generated by *B*.

Indirect evidence: If OBM *A* does not have direct evidence about OBM *B*, but if one of the other OBMs has confirmed that the block generated by *B* is valid, then *A* has indirect evidence about *B*.

Each OBM maintains a list that records pertinent information to establish direct evidence. For this, the OBM records the number of blocks it has validated for every other OBM. Since only the cumulative number of verified transactions is stored for each OBM, the associated memory overhead of the distributed trust table would be negligible. Recall from Section 2.3 that an OBM might create blocks which are non-compliant with the consensus algorithm. Other OBMs that receive a non-compliant block will drop it and decrement the direct trust associated with the responsible OBM by one. If the malicious OBM continues with this behavior, its trust rating would be correspondingly reduced. This implies that an increasing proportion of its transactions will have to be verified by the other OBMs. For indirect evidence,

Direct evidence	**Number of previously validated blocks**	10	20	30	40	50
	Needs to validate	80%	60%	40%	30%	20%
Indirect evidence	**Percentage of OBMs signed the block**	20%	40%	60%	80%	100%
	Needs to validate	80%	75%	70%	60%	40%

Fig. 2. An example of a trust table.

the OBM checks the number of other OBMs that have verified a received block generated by an OBM. The core idea behind the distributed trust algorithm is that the stronger the evidence an OBM has gathered about the OBM generating the new block, fewer transactions within that block need to be verified to validate the block. A trust table, an example of which is illustrated in Fig. 2 is maintained by the OBMs to implement this strategy. Direct evidence takes precedence over indirect evidence. If the OBM has direct evidence about the block creator, then a fraction of the transactions within the block are selected to be validated as per Fig. 2. In the case that there is no direct evidence, the OBM checks if indirect evidence is available and then selects a different fraction of transactions based on how many other OBMs have vouched for the block generator as per Fig. 2. Note that, a certain fraction of transactions are always verified even if there is strong evidence to protect against a potentially compromised OBM. If no evidence is recorded, then all transactions in the block are verified.

2.5. Distributed Throughput Management (DTM)

The classical consensus algorithms used in blockchain limit the blockchain throughput, which is measured as the number of transactions stored in the blockchain per second, as solving the cryptographic puzzle is computationally demanding. For instance, Bitcoin blockchain is limited to 7 transactions per second because of POW [38]. For IoT, such limits would be unacceptable, since there are numerous interactions (and thus transactions) among various nodes. Moreover, some of the transactions may require immediate actions (e.g. unlocking smart home door from overlay). In LSB we propose a Distributed Throughput Management (DTM) mechanism (outlined in Algorithm 2 below) to actively monitor the blockchain utilization and make appropriate adjustments to ensure that it remains within an acceptable range. At the end of every consensus-period, each OBM computes the utilization (α) as the ratio of the total number of new transactions generated to the total number of transactions added to the blockchain. Note that, since all transactions and blocks are broadcast to all OBMs, the utilization computed by all OBMs should be similar. The aim of DTM is to ensure that α remains within a certain desirable range (α_{min}, α_{max}).

Assuming a network with N nodes of which M are OBMs and R representing the average rate at which a node generates new transactions per second (R can be estimated from the total number of transactions generated in the consensus-period), the utilization can be represented as follows:

$$\alpha = \frac{N * R * Consensus\_period}{T\_max * M} \tag{1}$$

The above equation suggests that there are two ways by which the utilization can be adjusted: (i) changing the consensus-period, which dictates the frequency with which blocks are appended to the blockchain; or (ii) changing M, as each OBM can generate one block within the consensus-period. The latter approach incurs

Algorithm 2 Distributed Throughput Management.

Input: α
1: **while** true **do**
2: **if** ($\alpha > \alpha_{max}$) **then**
2: compute consensus-period$_{new}$ from Eq. (1) with $\alpha = \frac{\alpha_{min}+\alpha_{max}}{2}$
3: **if** (consensus-period$_{min}$ <= consensus-period$_{new}$) **then**
3: update consensus-period to consensus-period$_{new}$
4: **else**
4: reset consensus-period to default value
4: compute M from Eq. (1) with $\alpha = \frac{\alpha_{min}+\alpha_{max}}{2}$
4: recluster overlay
5: **end if**
6: **end if**
7: **if** ($\alpha < \alpha_{min}$) **then**
7: compute consensus-period$_{new}$ from Eq. (1) with $\alpha = \frac{\alpha_{min}+\alpha_{max}}{2}$
8: **if** (consensus-period$_{new}$ <= consensus-period$_{max}$) **then**
8: update consensus-period to consensus-period$_{new}$
9: **else**
9: reset consensus-period to default value
9: compute M from Eq. (1) with $\alpha = \frac{\alpha_{min}+\alpha_{max}}{2}$
9: recluster overlay
10: **end if**
11: **end if**
12: **end while**

significantly greater overheads as it requires reconfiguration of the entire overlay network (see Section 2). Thus, if α exceeds α_{max}, in the first instance, DTM checks whether the consensus-period can be reduced by checking if it has received the minimum consensus period. If so, then the new value for the consensus-period is computed using Eq. (1) and assuming that α is equal to the mid-point of the desired range (α_{min}, α_{max}), which ensures a stable operating point for the network (line 2–3, Algorithm 2). Conversely, if the consensus-period cannot be reduced then the network needs to be reclustered with a new value for M (line 4). This new value is computed using Eq. (1), with α again set to the mid-point of the desired range and the consensus-period set to the default value, which is *consensus-period*$_{max}$. This feature allows LSB to scale well, where an increased number of participating nodes delivers higher throughput. In order to avoid constant reconfiguration of the network when the utilization is increased above its threshold, we reset the consensus-period to default value. With a constant number of OBMs, the upper-bound throughput of LSB is primarily dependent on the value of the consensus-period which in turn depends on the end-to-end delay among the OBMs. If we neglect the end-to-end and synchronization delays, the throughput will be limited by the maximum rate at which the slowest OBM can generate new blocks which is the time taken to verify all transactions in a new block and collating them to form a block. This potentially prevents a resource-capable OBM (which may store blocks at a faster rate than other nodes) from gaining control of the blockchain. As can be inferred from Eq. (1), a smaller consensus-period leads to larger throughput.

In the instance when the utilization drops below α_{min} an inverse approach is adopted, i.e. DTM first attempts to increase the consensus-period, otherwise it decreases the number of OBMs (lines 7–9, Algorithm 2).

To ensure that all nodes are consistent about the action to be taken (whether it be changing the consensus-period or M), each OBM waits for a random duration and broadcasts a message specifying the action to be taken to all other OBMs. A recipient OBM checks whether the action is consistent with its decision. If so, it signs the original message and broadcasts it to other OBMs. If not, then it creates a fresh message specifying its action and broadcasts it to other OBMs. The action message that receives signatures from more than half the number of OBMs is assumed to be agreed-upon decision which all OBMs must follow. Note that, in most instances the actions taken by all nodes will be consistent. However, occasionally there may be slight discrepancies in the OBMs estimate of the number of generated transactions due to packet loss or latency issues, which may in turn lead to minor differences in the computed consensus-period or M. In the rare event that there is no clear majority, the OBMs employ an election method such as in [39] to reach a final agreement about the new consensus-period.

In the event where the number of OBMs are to be changed, the network is reclustered using the same clustering method used initially as discussed in Section 2.

2.6. Summary

This section provides a high-level view of all algorithms that are executed by an OBM for managing LSB. An OBM may either receive a transaction T or a block B from other OBMs. In the former instance, the OBM first verifies the transaction (by validating the embedded signature using the corresponding PK) and if valid adds it to the pool of pending transactions, T_Pool (line 3 in Algorithm 3). If the cumulative size of T_Pool equals or exceeds the block size, T_max, these transactions are collated to form a block, which is stored in the blockchain by invoking the consensus algorithm as discussed in Section 2.3 (line 4). An invalid transaction is discarded (Line 6). In the latter case, the OBM first verifies the block by verifying the constituent transactions (line 9). Recall that, LSB employs distributed trust algorithm (see Section 2.4) to reduce the processing overhead in verifying new blocks. If the block is valid, it is appended to the locally stored copy of the blockchain. Otherwise the block is discarded. The OBM is executing the DTM algorithm continuously in a parallel thread to manage the throughput of LSB in accordance with the network load (line 10).

Algorithm 3 A high-level view of algorithms executed by an OBM.

Input: T, B
1: **while** true **do**
1: **Receive From Blockchain**
2: **if** Received a T **then**
3: **if** (T is valid) **then**
3: T_Pool += T;
4: **if** (size.T_Pool >=T_max) **then**
4: Run DTC (see Section Section 2.3)
5: **end if**
6: **else**
6: Discard T
7: **end if**
8: **end if**
9: **if** Received a B **then**
9: Verify block (see Section Section 2.4)
10: **end if**
10: Run DTM (see Section Section 2.5)
11: **end while**

2.7. LSB compatibility

In this sub-section, we discuss LSB compatibility with: (i) existing blockchain solutions, and (ii) IoT requirements.

LSB introduces significant changes to core blockchain functions including: (i) the consensus algorithm (see Section 2.3), (ii) the underlying network topology, which is a clustered network (see Section 2.1), (iii) the separation of data and transaction flow (see Section 2.1), (iv) the distributed trust algorithm (see Section 2.4), and (v) distributed throughput management algorithm (see Section 2.5). LSB can rely on existing blockchains for providing other functionality, e.g., nodes that serve as miners in an existing blockchain could act as OBMs in LSB, or the peer-to-peer algorithms used in conventional blockchains can be used between OBMs.

LSB is designed particularly for IoT, and thus must fulfill the fundamental requirements of IoT, namely supporting real-time applications and the connectivity and mobility of IoT devices:

- Real-time applications: IoT devices are required to share data with the users or SPs in real-time that are used to offer real-time services. The distributed operations inherent in a blockchain incur delays and could thus potentially impact IoT applications that require real-time functionality. However, LSB incorporates a number of measures to reduce these delays and is thus suitable for real-time applications. These are: (i) Separation of data and transaction flow: The transactions are broadcast while the data packets are directly sent, i.e., routed, toward the destination. For example, when a camera receives an access transaction, it directly sends the data in real-time to the user. At the end of the data transmission, the corresponding transaction is stored in the blockchain that contains the hash of the data, (ii) Eliminating the confirmation time: In conventional blockchains, the blockchain participants have to wait at least 3 blocks to be chained to the block that contains the transaction to protect against double spending. As the concept of double spending is not applicable for communications between IoT devices, e.g., opening a smart lock, LSB does not demand the IoT devices to wait for transaction confirmation to accept a transaction. In cases where double spending is applicable, e.g., trading asset, the delay for confirmation will still apply in LSB.
- Connectivity and mobility: In IoT the connectivity of a node cannot always be guaranteed. Additionally, the nodes might be mobile nodes that change their position in the network. We discuss these from the perspective of the cluster members and the OBMS.
  Mobility or connectivity issues of the cluster members do not affect the blockchain state and function as the blockchain is managed by OBMs. A node that is disconnected no longer receives service as in other existing systems. A mobile node can continue receiving service from its OBM, however, the experienced delay in communication with the OBM may increase. To address this challenge and reduce the delay while maintaining the nodes connectivity, LSB leverages existing soft handover methods such as in [29]. The mobile node joins a new OBM that has lower delay compared to its older OBM and updates its keys in the new OBMs key list. Once the join request is accepted by the new OBM, the mobile node disconnects from its previous OBM and removes its keys from the key list of that OBM. Thus, the mobile node can change its OBM with no interruption in service.
  Next, we discuss mobility and connectivity from the perspective of the OBMs. Note that OBMs are typically selected from the nodes with no or very low mobility and high connectivity. In case an OBM is disconnected from the network, the cluster members and blockchain throughput will be temporarily affected. Note that exiting clustering algorithms choose one node as the leader [31], i.e., OBM in LSB, and another node as the co-leader. Once the leader leaves the network, the co-leader will become the new leader so that the cluster members will not be disconnected from the network for a long time to find a new leader.
  The network throughput is also affected as there are fewer OBMs to generate new blocks. Recall from Section 2.5 that the DTM manages the throughput by measuring the network utilization or α (see Section 2.5). α shall always meet the following condition: $\alpha_{min} <= \alpha <= \alpha_{max}$
  The network throughput is not affected by departure of an OBM if α remains within the specified range. If it exceeds the threshold, the DTM adjusts the throughput.

3. Case study

In the previous section, we outlined LSB and its key features including low resource consumption and high throughout, that are suitable for a broad range of applications. To ground the discussion and evaluation, we focus on the case study of a smart home.

We assume the smart home is equipped with a number of IoT devices, e.g., smart thermostat, smart lock, smart light, etc. IoT devices with significantly varying capabilities are now available on the market, e.g. Amazon Echo [11] which has high resources and motion sensors [45] which have more restricted resources. The high resource IoT devices, which are at the high end of the spectrum, can readily handle asymmetric encryption [11] while the devices at the other end of the spectrum can only afford symmetric encryption. Thus, we study two distinct use cases of LSB in the smart home that covers both these types of devices in Sections 3.1 and 3.2.

As discussed in Section 1, storing data of IoT devices in blockchain is not scalable and incurs significant overheads. LSB offers flexibility whereby the data of IoT devices can be stored in: (i) Local storage: This storage can be integrated with the home Internet gateway or it can be a separate backup device. IoT devices use direct communication to store data in the local storage as outlined in Sections 3.1 and 3.2, (ii) Cloud storage: We assume that a user who wishes to store the data of his IoT devices in the cloud has created an account with a cloud storage provider (e.g., Dropbox, OneDrive, etc.) out-of-band (i.e., independent of LSB). We assume that the user creates a public/private key pair for this cloud storage account and that the corresponding public key is used in subsequent store cloud and access transactions. Recall that LSB creates a clear distinction between the control plane and the data plane to ensure that the data packets can be routed efficiently through the network. To facilitate this, we assume that the IoT device that requires to store data in the cloud sends a request during the initial setup to the cloud storage with the aforementioned PK. Upon authentication, the cloud storage sends the ID of its OBM to this device. Subsequently, all data being stored in the cloud can be directly routed to the cloud (as discussed in Section 2.1).

Having discussed the basic concepts of the smart home, we study two scenarios based on the resources available for the IoT devices in the rest of this section.

3.1. Scenario 1: High resource devices

In this scenario, we assume that IoT devices are capable of performing asymmetric encryption. Each IoT device is part of the overlay and is known by a PK. The IoT devices within the smart home can directly communicate with each other, i.e., off-the-chain communication, using the existing communication protocols including Wi-Fi or ZigBee. As these communications take place within the smart home, no record is required to be stored in the blockchain. In contrast, when IoT devices communicate with overlay nodes, corresponding transactions are stored in LSB to record these interactions. In the following, we outline the process of store, monitor, and access transactions. As in Bitcoin and other blockchain-based systems where a transaction generator knows the PK of the transaction receiver, e.g., the node that should be paid in Bitcoin, it is assumed that the requester knows the PK of the requested IoT device.

Store: An IoT device might require to store its data either in the local storage or in a cloud storage. As outlined above, the IoT device can store data locally by encrypting data with the PK of the local storage and directly communicating with the storage. An IoT device may also need to store data in a cloud storage. The

flow of events for storing data in the cloud is shown in Fig. 3. The IoT device that wishes to store data in the cloud sends data with the OBM ID of the cloud to its own OBM (S1 in Fig. 3). The OBM then routes data directly to the cloud storage using routing protocol. After storing data, the cloud storage signs the received transaction from the IoT device and sends it to its own OBM to be stored in the blockchain (S2).

Access and monitor: The flow of access and monitor transactions are shown in Fig. 3. For both transactions, the requester, i.e., the node that is requesting IoT device's data, generates and sends a multisig transaction, which requires 2 out of 2 signatures to be considered as valid, to its OBM (A1, M1 in Fig. 3). The OBM checks the keylists to find a match and if not then broadcasts the transaction to other OBMs (A2, M2). By finding the match, the OBM forward the transaction to the IoT device (A3, M3). The fourth step differs for access or monitor transaction. For access transaction, the data is fetched from either the local or cloud storage (A4), while for monitor transaction the device sends the real time data to the requester (M4). After receiving data from either cloud or local storage, the IoT device routes it to the requester (A5). Recall that the data flow is routed directly and separate from the transaction flow. Finally, the IoT device signs the received transaction from the requester and sends it to its OBM to be stored in the blockchain (A6, M5).

3.2. Scenario 2: Low resource devices

Next, we consider how low resource IoT devices interface with LSB. Performing asymmetric encryption might be far beyond the capabilities of these devices. Thus, we introduce a central controller in the smart home which connects these IoT devices to the overlay network as shown in Fig. 4. Local communications, i.e., the communications between IoT devices, central controller, and the local storage, are encrypted using symmetric encryption, for which a shared key is established between the two parties, and use lightweight cryptographic hash function, such as in [13]. The central controller could be integrated with the Internet gateway or a stand-alone middlebox such as F-secure [43] which acts as an intermediary between the IoT devices and the gateway. The central controller uses the generalized Diffie–Hellman [21] key distribution method to generate and distribute a shared key between two local entities that request to exchange data.

In the following, we outline the process of store, monitor, and access transactions.

Store: Similar to scenario 1, the IoT devices may store data either locally or in the cloud storage. In the former instance, the device sends its store request to the central controller. The central controller generates and distributes a shared key between the device and the local storage. Local storage uses the shared key for authentication. For further communications, the device and the storage communicate directly using the shared key.

The flow of events for storing data in the cloud are shown in Fig. 4. The IoT device sends its data to the central controller (S1 in Fig. 4). This data is encrypted with the shared key of the device and the central controller. The central controller forwards the data to its own OBM with the destination address corresponding to the OBM ID of the cloud storage. As noted earlier, the data is directly routed to the cloud storage (S2). Similar to scenario 1, the OBM of the cloud storage signs the transaction and stores it in the blockchain (S3).

Access and monitor: The flow of access and monitor transactions are shown in Fig. 4. The first 3 steps are as scenario 1. However, unlike scenario 1 where the requester sends a request directly to the device, in scenario 2 the requester sends the request to the central controller of the smart home in which the device is located. The ID of the requested device is populated in the metadata field of the transaction (see Section 2.2). The fourth step differs for access or monitor transaction. For access transaction, the central controller fetches data from either the local or cloud storage (A4), while for monitor transaction the central controller requests the real-time data of the device (M4). After receiving data, the central controller routes it to the requester (A5, M5), then signs the received transaction and stores it in the blockchain (A6, M6).

4. Evaluation and discussion

In this section we provide qualitative security and privacy analyses as well as quantitative performance evaluation.

4.1. Security and privacy analysis

In this section, we discuss LSB security, privacy, and fault tolerance.

Threat Model: It is assumed that the adversary (or cooperative adversaries) can be the OBM or a node in the overlay network. Adversaries are able to sniff communications, discard transactions, create false transactions and blocks, analyze multiple transactions in an attempt to deanonymize a node, and sign fake transactions to legitimize colluding nodes. The adversary can pretend to be multiple nodes as well as OBMs by generating transactions or blocks with multiple PKs to flood the network or consume the resources of the participating nodes. Adversaries can collude to compromise the security of the consensus algorithm by ignoring the frequency with which an OBM is generating new blocks. We assume that standard secure encryption methods are used in the overlay, which cannot be compromised by adversaries.

Security: Table 2 summarizes the various mechanisms that allow LSB to meet key security requirements.

In following we study seven specific security attacks to which IoT networks or blockchains are particularly vulnerable and outline how LSB protects against them.

Denial Of Service (DOS) attack: In a DOS attack, the attacker floods and overlay node (target) with a large number of multisig transactions (which require the target signature) to overwhelm the node such that it cannot devote any resources to process genuine transactions from other nodes. LSB protects against this attack using the following defense methods (see Section 2): (i) OBMs would not send a transaction to their cluster members unless they find a match with an entity in their key list, (ii) Each overlay node has a threshold for maximum rate of transactions received from the overlay. If the threshold is exceeded, the keylist is updated to prevent nodes from sending transactions to the target node.

Distributed DOS (DDOS): This attack is the distributed version of DOS attack where multiple overlay nodes are compromised by the attacker. The methods discussed to protect against DOS attack also protect again DDOS attack. Additionally, infecting devices and overlay nodes is difficult due to usage of OBM keylists (see Section 2).

Dropping attack: In this attack, the OBM drops transactions to or from its cluster members to isolate them from the overlay. To protect against this attack, a cluster member can change the OBM it is associated with if it observes that its transactions are not being processed.

Blockchain modification: The attacker advertises a false ledger of blocks and makes it as the longest ledger. Thus, all nodes accept the attacker ledger as the true ledger. The proposed DTC algorithm (see Section 2.3) limits the number of blocks each OBM can generate within a time interval. This will limit the number of malicious blocks that an OBM can append, and thus prevent the attacker from generating a longer ledger than the true ledger.

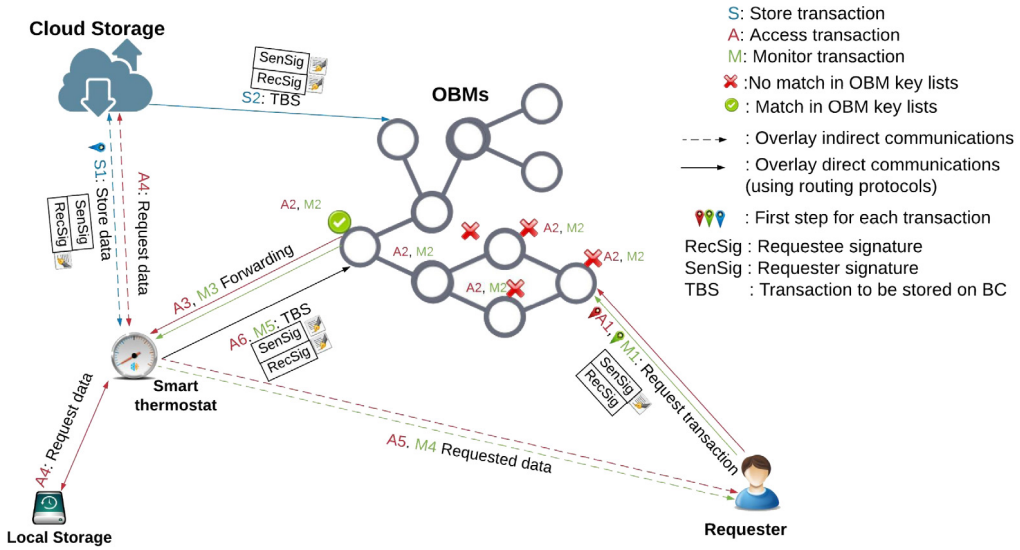

Fig. 3. The process of store, access, and monitor transactions for high resource devices.

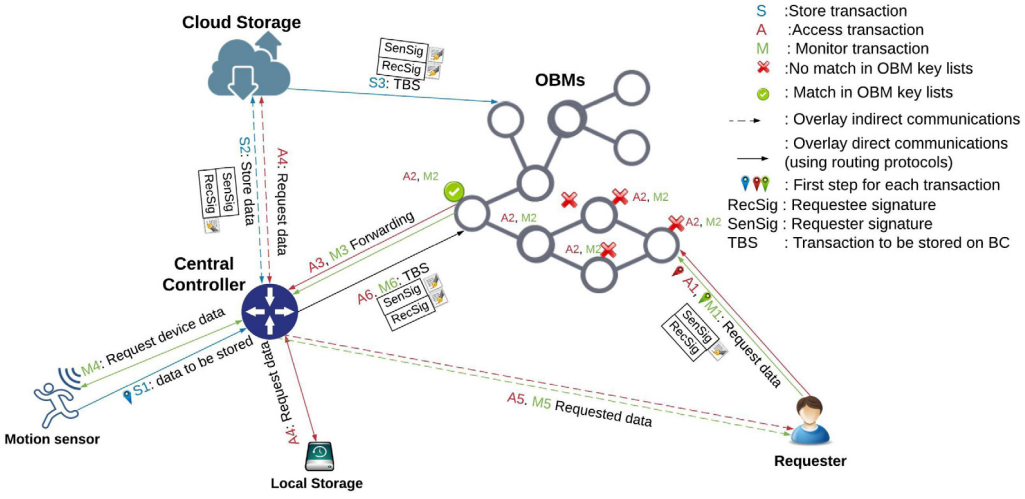

Fig. 4. The process of store, access, and monitor transactions for low resource devices.

Table 2
Security requirements discussion.

Requirement	Employed method
Confidentiality	Encryption is used for all transactions (Section 2).
Integrity	Each transaction includes a hash of all other fields contained in the transaction (Sections 2).
Availability	An OBM sends a transaction to its cluster members only if a key contained in the transaction matches one of the entries in its keylist (Section 2). This ensures that the cluster members only receive transactions from authorized nodes.
Authentication	Each node should have a stored genesis transaction in the BC to be authenticated. As transactions are chained to the genesis transaction, a node is authenticated when it has the private key corresponding to the output PK of a transaction stored in the BC (Section 2).
Non-repudiation	Transactions are signed by the transaction generator to achieve non-repudiation. Additionally, all transactions are stored in the BC, so involved parties in the transaction can deny their complicity in a transaction (Section 2).

Compromising the time interval: In this attack, a malicious OBM generates more than one block in each consensus-period. This attack can be conducted by a group of collaborative malicious nodes. Multiple OBMs will verify that a malicious block follows the consensus rules and thus can be appended to the blockchain. To prevent this attack, each OBM stores the time-stamp of the last block that other OBMs in the network have generated. Thus, each OBM independently decides whether an OBM follows the consensus rules. If the time difference between two blocks generated by the same OBM is less than the consensus period, the new block will be rejected. The time-stamp is maintained and updated along with the trust table.

Compromising the waiting time period: In this attack malicious OBMs collude to generate blocks with short waiting time to store fake transactions in the blockchain. Recall from Section 2.3 that each OBM waits for a random time before generating a new block. The neighboring OBMs monitor the random time to ensure that an OBM does not always select a short waiting time. However,

a group of malicious neighboring nodes may collude and always allow a selected node(s) to choose short waiting times and subsequently generate blocks. This attack can be detected by the honest OBMs during multiple rounds of consensus period when the malicious OBMs mine multiple blocks with short waiting time. By monitoring the waiting time of multiple blocks generated by each of the malicious nodes, the honest OBMs detect the attack. Even if the attackers succeed, the security of the blockchain remains unaffected as still the malicious OBMs cannot generate more than one block per consensus period (see Section 2.3). Moreover, the OBMs verify the transactions in new blocks, thus will detect any fake transaction stored in the blockchain. In summary, this attack provides no advantages for the malicious nodes that worth it to allocate resource and effort for conducting the attack.

Sybil attack: In Sybil attack a malicious node pretends to be multiple nodes by creating multiple identities. The malicious nodes can add fake transactions and in the worst case get control of the blockchain. We study Sybil attack from the perspective of a node in the blockchain and an OBM.

In LSB, each participant must have a previous transaction in the blockchain to chain its new transactions which implies the need to have a genesis transaction to begin this transaction chain. Recall that a genesis transaction is the first transaction in each ledger. In LSB the generation of genesis transaction either requires the user to burn coin in Bitcoin or receive a certificate from trusted CAs, both of which incur monetary costs (see discussions in Section 2). The creation of multiple identities thus requires significant investment from the attacker. The transactions generated by nodes are verified by the OBMs, thus any misbehavior will be detected by the OBMs. In case of a successful Sybil attack by participating nodes, the OBMs utilize the proposed DTM algorithm (see Section 2.5) to balance the throughput to mitigate the delay experienced by the end-users for receiving service. However, additional OBM resources will be invested to generate and verify blocks.

An OBM may also pretend to be multiple OBMs to generate more blocks. Once OBMs are selected, everyone will be notified of the PK of the OBMs. Thus, malicious OBMs cannot pretend to be multiple OBMs as the blocks generated with a PK that is not listed in the OBM list will be rejected by other OBMs. However, an OBM may attempt to generate fake identities and create fake OBMs during network clustering. The OBMs are selected distributively during the clustering phase. The nodes with maximum number of cluster members are chosen as the CHs. The malicious node cannot fake one hop communication with a large number of participants, which protects against Sybil attack. To further increase the security, we assume methods such as in [52] are in place to protect against this attack. The blocks generated by the OBMs are verified by participating nodes which further protects against this attack.

Appending attack: In this attack, a malicious OBM attempts to store fake transaction(s) in blockchain. Recall from Section 2.4 that the OBMs verify transactions in a block that protects against this attack. However, LSB employs distributed trust algorithm where fewer percentage of transactions in new blocks are verified as OBMs build up trust. Since only a fraction of transactions within a block are verified, there is a chance that a fake transaction may not be verified and thus appended to the blockchain. Based on the simulation results given in Fig. 8 a minimum of 13 OBMs must be active in the network to ensure blockchain resilience against appending attack. We will further elaborate on this attack in Section 4.2.

Consensus period attack: In this attack, the attacker sends false requests to update the consensus period. In LSB, for a request to be considered as valid, it must be signed by at least half the number of OBMs. The likelihood f this is very low.

Table 3
Studying attacks on LSB.

Attack	Resistant to attack	Attack likelihood
Denial Of Service (DOS) attack	High	Unlikely
Distributed DOS (DDOS) attack	Beyond high	Unlikely
Dropping attack	High	Unlikely
Blockchain modification	Beyond high	Unlikely
Compromising the time interval	Beyond high	Unlikely
Compromising the waiting time period	Moderate	Possible
Sybil attack	Beyond high	Unlikely
Consensus period attack	High	Unlikely
51% attack	Beyond high	Unlikely

51% attack: The attacker controls more than 51% of OBMs and tries to compromise the consensus algorithm by generating fake blocks or more than the permitted number of blocks. This attack can be detected during the block verification or by other OBMs based on the consensus algorithm.

In Table 3 we analyze how resilient LSB is against the aforementioned attacks and the likelihood of the attack to happen based on European Telecommunications Standards Institute (ETSI) [47] risk analysis criteria. These criteria evaluate each attack based on the following five metrics: (i) *time:* the cumulative time for an attacker to first detect a vulnerability, and subsequently plan and launch a successful attack, (ii) *expertise:* the generic expertise that the attacker must possess about the underlying principles in order to orchestrate the attack, (iii) *knowledge:* specific information that is available about the target system, e.g., security configuration, (iv) *opportunity:* the duration and nature (e.g., continuous or intermittent) of access to the system needed for launching the attack, (v) *equipment:* software and/or hardware necessary for conducting the attack. LSB exhibits beyond high resistance to five attacks and high resistance to three attacks. This suggests that LSB is highly secure.

We now discuss the adversary tolerance level for LSB and compare with existing blockchain consensus algorithms. The adversary tolerance level represents the blockchain resilience level against malicious behaviors. The adversary tolerance level for LSB is 51% of the participating nodes. In other words, as long as 51% of the participating nodes are honest, the security of the ledger can be guaranteed as the honest nodes can generate the longest ledger. Recall from Section 2.4 that LSB uses distributed trust to reduce the processing overhead for verifying new blocks. The adversary tolerance level for distributed trust depends on the trust table values. Fig. 7 and Table 4 demonstrate the simulation results for adversary tolerance based on the adopted trust table values for our experiments.

In this paragraph we discuss the adversary tolerance level for PoW, Byzantine Fault Tolerance (BFT), and Proof of Elapsed Time (PoET) for comparison with LSB. Details of these algorithms are discussed in Section 6. The adversary tolerance level identifies the extent to which a consensus algorithm can tolerate malicious activities.

The adversary bound for PoW is 51% of the computational, i.e., mining, power [50]. Malicious nodes may be able to compromise the security of the PoW-based blockchain if their combined computational power reaches at least 51% of the blockchain computational power. In BFT algorithm the validator, i.e., miner, of the next block is chosen based on voting of the nodes to the new blocks. The adversary tolerance level for BFT is 33% of the participating nodes. PoET is a time-based consensus algorithm that enforces the validators to wait for a random time prior to adding new blocks to the blockchain. PoET uses Intel hardware to enforce the randomness. The authors in [15] argued that the adversary tolerance level for PoET is $\theta(\frac{loglogn}{logn})$.

In summary, LSB achieves higher adversary tolerance level compared to BFT and PoET. The adversary tolerance level of LSB

cannot be compared with PoW as it is based on the number of participants in LSB while in PoW it is based on the computational power of the adversaries.

Privacy: LSB uses anonymity and affords users control over their data to protect their privacy. Similar to Bitcoin, each user may employ multiple identities, i.e., PKs, for his transactions which introduces high-level of anonymity. LSB empowers the users to control access to their devices and data. Recall that blockchain achieves high auditability by permanently storing all communications in the public immutable ledger. Thus, LSB users can monitor the data transferred by their devices to detect any malicious activity.

In certain IoT applications, the two end points that are communicating may need to know the real identity of each other. For example, a home insurance company needs to know the real identity of the owner of the smart home that it is insuring. In these instances, the corresponding transaction generator uses a unique PK to communicate with each overlay node.

Malicious nodes may attempt to deanonymise a user by tracking his transactions stored in blockchain which in turn endangers the user privacy. This attack is known as "linking attack" in blockchain literature. Linking attack is possible in all blockchain instantiations and LSB is not an exception. To protect against linking attack, a user may employ multiple ledgers for each of his devices and change his key for each transaction.

Recall that transactions in blockchain reflect communications between IoT devices and/or SPs. Thus, an adversary may be able to collect activity patterns, even though the transaction metadata is encrypted. However, such attacks are inherent in all blockchains (not just LSB) as all transactions and blocks are broadcast to participating nodes. To reduce the success rate of this attack, the participating nodes may employ multiple ledgers in LSB to store their transactions. This attack is well-studied for cryptocurrencies but there are few works to discuss this attack for IoT device identification. In [23] we have studied the success rate of collecting device activity pattern in existing blockchain instantiations using machine learning algorithms. The implementation results demonstrate that the attacker can successfully classify data with 90% accuracy.

Fault tolerance: Fault tolerance is a measure of how resilient an architecture is to node failures. It is evident from Section 2 that OBMs implement various key functions and the failure of these nodes could thus potentially impact the normal operation of LSB. In case an OBM leaves the overlay, the cluster members associated with this OBM would not receive any service. However, they can readily select a new OBM to associate with. The OBM(s) departure may also affect the overlay throughput as there are fewer OBMs to generate blocks. However, the DTM mechanism outlined in 2.5 can handle this situation. The departure of multiple OBMs may also impact security due to the corresponding actions of the distributed trust mechanism. Recall from Section 2.4 that as OBMs garner trust in one another, fewer transactions within a block need to be verified. Thus, when multiple OBMs leave, the probability of detecting a fake transaction in a new block decreases as fewer OBMs remain in the network to validate the new blocks. We will further elaborate on the minimum number of OBMs required to participate in the blockchain to prevent attacks in Section 4.2.3.

4.2. Performance evaluation

In this section, we present extensive evaluations of various performance aspects of LSB. We first explored the possibility of using open source blockchain instantiations such as Ethereum. However, these platforms are particularly suited for developing applications on top of the underlying blockchain substrate. LSB has significant differences in its fundamental operations in comparison to these blockchain instantiations. As a result, we were unable to use these platforms for our evaluations and thus chose to use simulations. We use NS3 [4] to evaluate the performance as it has been widely used for analyzing peer-to-peer networks.

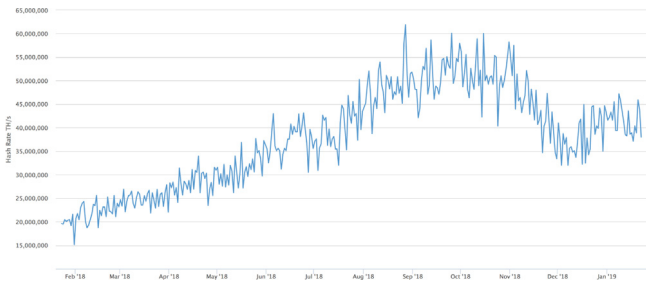

Fig. 5. Bitcoin blockchain hash rate as at Jan 2019 [6].

We consider a network consisting of 50 overlay nodes. We assume the *T_max* to be 10 transactions. We assume five overlay nodes generate four transactions per second. The above settings are referred to as the default configuration and are used in the simulations unless explicitly noted otherwise.

In the rest of this section, we first evaluate the POW processing time in Section 4.2.1. Next, we evaluate the delay which an overlay node experiences while requesting data from another overlay node in Section 4.2.2. Distributed trust and its effects on the overlay security and performance are studied in Section 4.2.3. Finally, we evaluate DTM in Section 4.2.4.

4.2.1. POW processing time

In this part of evaluation, we aim to evaluate the time consumed by an off-the-shelf device to solve the POW, one of the widely used consensus algorithms in blockchain-based systems [25]. We do so to highlight the ineffectiveness of using classic blockchain and PoW in the IoT context. Each block in the Bitcoin blockchain has a nonce attached to them. The miner is required to search for the correct nonce such that the block as a whole satisfies a certain arbitrary condition. Specifically, it is required that the SHA-256 hash of the block has a certain number of leading zeros. The only way to find the correct nonce is by brute force. The number of leading zeros controls the difficulty of solving the POW. The longer the length of this sequence, the more resources and processing time required to solve the puzzle.

We study the possibility of using laptop to solve PoW puzzle using MinerGate [8]. We used a MacBook Pro (2.7 Ghz Intel Core i5 processor, 8 GB memory, and Intel Iris Graphics 6100 graphic card) for this study. Typical IoT devices are significantly more resource-constrained than a laptop so the results obtained are conservative upper bounds that one can expect with IoT devices. Once running the MinerGate, the hash rate reached maximum of 1 KHs. The laptop, however, fails mining a block in one day. This is expected as currently the mining hash of Bitcoin blockchain is around 38,000,000 THps [6] (see Fig. 5). Given the difficulty of Bitcoin blockchain, mining requires specific miner hardwares known as Application-Specific Integrated Circuit (ASIC) miners with at least Trillion hashes per second.

4.2.2. Requesting an IoT device

In this section, we evaluate end-to-end delay experienced by an overlay node for requesting the data of an IoT device which is performed using multisig transaction. Delay is measured from the time since the request is generated till the response is received. Note that in this evaluation, we only consider the request/response delay, while data exchange delay is studied in

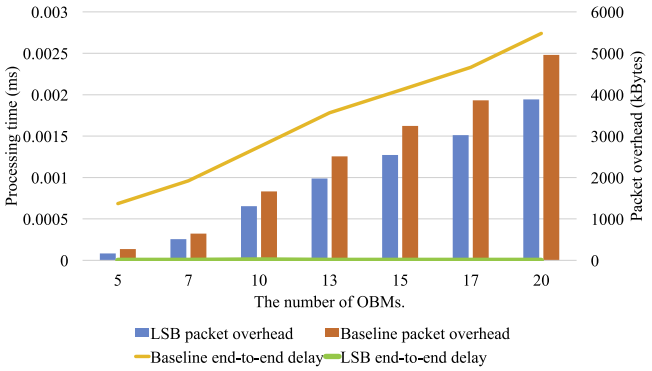

Fig. 6. Assessing the impact of separating the data and transaction flows.

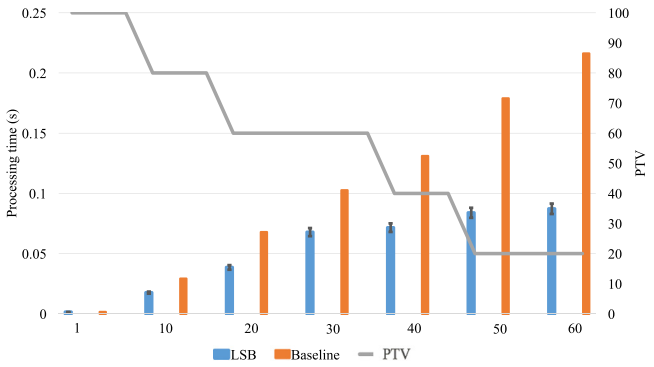

Fig. 7. The average processing time on OBMs to validate new blocks.

the next evaluation. We conduct simulations using NS3 with the default configuration with 13 overlay nodes acting as OBMs. We compare LSB with a baseline method which is consistent to current methods offerings on the market, where the two nodes directly communicate without the need for any of the transaction processing that is part of LSB. The delay incurred using the baseline method is 17.62 ms. On the contrary, with LSB, the delay increases to 48.74 ms. The higher delay can be attributed to the fact that the transaction has to be broadcast to other OBMs for verification. Each OBM incurs a delay of 0.006 ms for processing the transaction (the precise steps are outlined in Section 2). However, this delay is relatively insignificant.

As was noted in Section 2, LSB separates the data flow from the transaction flow. While transactions are broadcast amongst the OBMs in the overlay, the data packets are forwarded toward the destination along optimal paths as determined by a routing protocol such as OSPF. To quantify the benefits of this design decision, we compare LSB with a baseline method wherein both the transactions and data packets are broadcast in the overlay network. We use the default configuration and assume that an overlay node (requester) sends four multisig transactions per second to a another overlay node (requestee). We consider the following two performance metrics which are best at capturing the impact of the separation between the transaction and data flows: (i) end-to-end delay — similar to above (ii) packet overheads — this captures the total number of packets transmitted by OBMs for delivering the data packets to the requester. Since, the size of the data and transaction packets are different, we measure the latter as the cumulative sum of all packet sizes in KBytes. Since, these two metrics are affected by the number of OBMs in the network, we vary the number of overlay nodes that act as OBMs from 5 to 20. The results are presented in Fig. 6. LSB incurs lower packet overhead and end-to-end delay compared to the baseline since, in the latter the data packets are broadcast among all OBMs as compared to the former where the data packets are routed along optimal paths. Observe that, for the baseline, both metrics grow linearly as the number of OBMs increases. The amount of broadcast traffic generated is directly proportional to the number of OBMs which explains the linear increase in the packet overhead. Since the data packets are now broadcast, the delay incurred in receiving the data at the requester also increases linearly. In contrast, with LBS only the packet overhead increases with the number of OBMs. Since the data packets are routed directly to the requester, the end-to-end delay is not affected by the number of OBMs.

These results demonstrate the efficacy of keeping the data and transaction flows independent of each other.

4.2.3. Evaluation of the distributed trust algorithm

Recall that in the classical blockchain, all transactions within a new block must be verified by an overlay node. In contrast, LSB uses a distributed trust algorithm wherein the number of transactions that must be verified decreases gradually as OBMs build up trust in each other (see Section 2.4). In this experiment, we compare the processing time for validating a new block in LSB with a baseline strategy that is similar to classical blockchains. We use the default network configuration and the trust table shown in Fig. 2. The simulation lasts for 180 s and the results, shown in Fig. 7, are the average of 10 runs. The standard deviation is also shown, except for the baseline where results are deterministic. We measure the time taken by each OBM to validate a new block and plot the average in Fig. 7 (shown on the left vertical axis). Note that, we disregard all other tasks (e.g. checking key lists, generating new blocks, etc.) other than validation of new blocks in this evaluation as the former are not affected by the trust algorithm. Fig. 7 plots the processing time as a function of the number of blocks successfully verified (and thus appended to the blockchain) as the simulation progresses. The percentage of transactions that need to be verified (PTV) is shown on the right vertical axis. As can be inferred from Fig. 7, at start up, the processing time is the same for both methods since the OBMs have yet to garner trust in each other. However, as time progresses and more blocks are generated and verified, the OBMs build up direct trust in each other. Consequently, only a fraction of the total transactions in a new block need to be verified in LSB, which reduces the processing time as compared to the baseline, wherein all transactions within the block are verified. Moreover, as the number of blocks verified increases, progressively less transactions need to be verified (also shown in Fig. 7) as the trust in other OBMs continues to increase. Once 50 blocks are generated, the trust among OBMs reaches the highest level (see Fig. 2). From here on, the number of transactions that need to be verified remains fixed and so does the processing time. At steady state (i.e., when the network has been running for a substantial period of time), LSB achieves over 50% savings in processing time compared to the baseline.

In LSB, since only a fraction of transactions within a block is verified, there is a chance that a fake transaction created by malicious node may not be verified and thus appended to the blockchain (referred to as appending attack in Section 4.1). In the following, we evaluate the success percentage of such an attack. Intuitively, the more OBMs in the network, the lower the likelihood of a successful attack, since the chance that the fake transaction will be picked for verification increases. However, the packet overhead also increases proportionally with the number of OBMs due to the increase in the broadcast traffic. To study this trade-off, we consider the default network configuration and vary

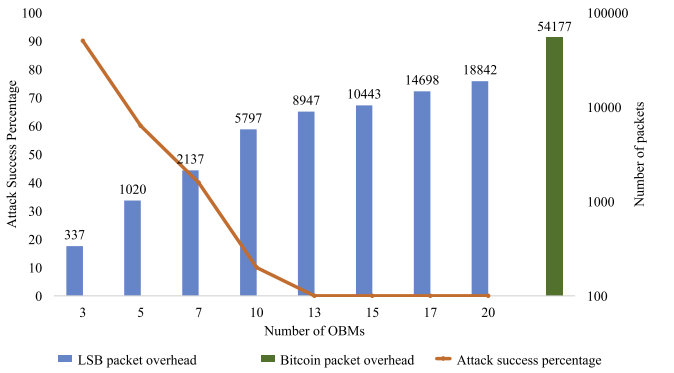

Fig. 8. Evaluating the impact of the number of OBMs on security and packet overhead.

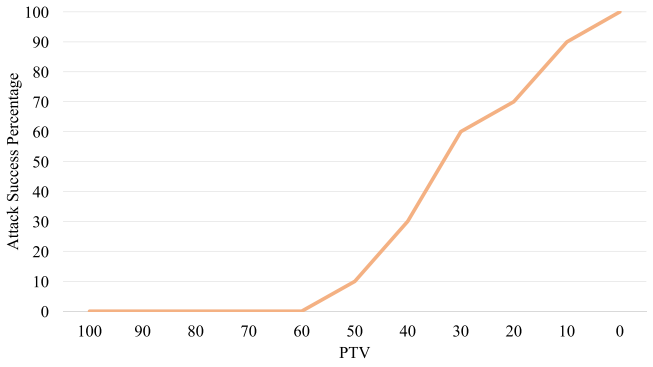

Fig. 9. Evaluating the impact of PTV on the ability to detect appending attacks.

the number of overlay nodes acting as OBMs from 3 to 20. The evaluation metrics are the attack success rate and the cumulative packet overhead. To simulate the attack, we consider the worst-case scenario, where a highly trusted OBM, which has generated more than 50 blocks and has thus accrued a high level of trust, creates a new block containing one fake transaction. We use the trust table shown in Fig. 2. We run the simulation 10 times and attack success is the percentage of the number of runs that the fake block is not detected by any of the honest OBMs (this applies to all evaluations that consider security attacks in the rest of the paper). We compare the packet overheads incurred in LSB with a baseline wherein the overlay network is structured similarly to Bitcoin. Recall that in Bitcoin all overlay nodes (50 in our case) manage the blockchain distributedly unlike LSB where blockchain management is limited to selected overlay nodes, i.e., OBMs. Note that, the baseline would always accurately detect the attack, since all transactions in a block are verified. The results are shown in Fig. 8. Observe that, as the number of OBMs increases, the likelihood of a successful attack reduces substantially. As expected, the packet overhead increases linearly with the number of OBMs. With 13 OBMs, all attacks are successfully detected. However, the corresponding packet overhead (8497) is significantly lower than that incurred in the baseline (54177).

The attack success percentage is directly impacted by the PTV. To study this impact, we evaluate the attack success percentage for different PTVs in a network with default configuration with five overlay nodes acting as OBMs. The reason for choosing five OBMs is to show the effect of PTV on the attack success. As is evident from Fig. 8, the presence of a greater number of OBMs improves the security considerably and these effects are not as evident. The results are illustrated in Fig. 9. When PTV equals 100, OBMs verify all transactions in a block, leading to zero attack success percentage (i.e., the attack is always detected). As shown in Fig. 9, as PTV decreases, the attack success percentage increases. For the network configuration used in this simulation, the lowest value of PTV that can guarantee security is 60. We have repeated the same simulation for different number of OBMs to determine the smallest value of PTV for which an attack can always be detected. Table 4 shows the results of the simulation and can be used as a guideline to configure the trust table (e.g. Fig. 2). A PTV that is lower than the values in Table 4 will make the network vulnerable to appending attacks. On the other hand, a larger value increases the processing time for new blocks (as more transactions need to be verified) and the packet overhead in the network. A detailed study on determining the trust values and its impact on blockchain security is provided in [20].

Table 4
Minimum PTV for detecting appending attacks as a function of the number of OBMs.

Number of OBMs	3	5	7	10	13	15	17	20
Minimum PTV	80	60	60	40	20	20	20	10

4.2.4. DTM performance analysis

The DTM mechanism proposed in Section 2.5 aims to dynamically adjust the network utilization based on the total load, i.e., the number of generated transactions. To illustrate the performance of DTM, we simulate a network with the default configuration with 13 overlay nodes acting as OBMs. As classical blockchains have fixed throughput (e.g., the Bitcoin blockchain has a fixed throughput of 7 transactions per second) there is no baseline that we can use for comparison. Initially, a total of 10 cumulative transactions are generated in the overlay network per second. We simulate situations where the network demand fluctuates. The number of transactions per second increases to 32 for the entire time period from 5 s to 40 s. At 40 s, the load increases further to 44 transactions per second until 45 s when the load drops to 12 transactions per second. The changes in the network load are illustrated in Fig. 10. Recall that DTM computes the network utilization, α, at the end of each consensus-period as the ratio between the number of transactions generated and the number of transactions added to the overlay blockchain since the last computation of α. Time intervals when α is computed by OBMs are shown using gray dots in the figure. The consensus period is initially set to 10 s, which is the default value. We assume that α_{min} and α_{max} are set to 0.25 and 1, respectively.

At the end of the first consensus-period (i.e. 10 s), α is computed to be 2.4, which is greater than α_{max} (1). This is because of the sharp increase in the network load at 5 s. To reduce the network utilization, DTM reduces the consensus-period to the newly computed value of 2.5 s (see lines 2–5 Algorithm 2) using Eq. (1), where α is set to 0.62, which is the mid point of α_{min} and α_{max}. The consensus-period is also illustrated in Fig. 10. Subsequently, since the network load remains stable until 40 s, the consensus period also remains unchanged. At this time, the network load increases further. Thus, at the end of the next consensus-period (43 s), α is computed to be 0.84. As the computed value is still less than α_{max}, no further action is required. This highlights the effectiveness of choosing the mid-point of α_{min} and α_{max} for recomputing α. The value of α drops to 0.2 at 48 s resulting from the sharp decrease in the number of transactions at 45 s. Since this value is less than α_{min}, DTM increases the consensus-period (see lines 7–9 Algorithm 2) to a new value, which is computed as 7 s.

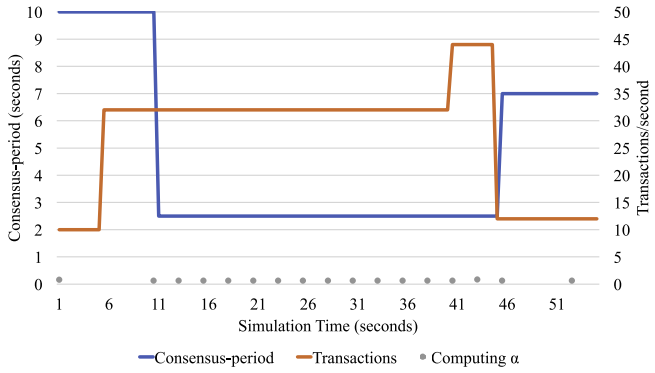

Fig. 10. Evaluation of DTM in the overlay.

5. Discussion

5.1. OBM reward

In classical blockchains, e.g. Bitcoin or Ethereum, nodes that generate new blocks are offered a monetary reward in the form of coins as a form of compensation for expending their resources to solve the computationally intensive puzzle associated with block creation. This fee is paid as the transaction fee by the users. However, there is now a growing consensus that for more effective blockchain the transaction fee should be removed [55]. LSB employs a lightweight consensus algorithm and thus we do away with explicit rewards and the transaction fee. Instead, an OBM on generating a valid block gains reputation with other OBMs (see Section 2), which could be construed as an implicit reward.

Another way to incentivize OBMs could be to allow them to place advertisements in the blocks that they append to the blockchain. An explicit field within the block header could be reserved for this purpose. The advertisements are displayed in the wallet software and allow the entities acting as OBMs to publicize their product that eventually results in increased revenue. The tasks performed by OBMs for managing the blockchain including forming new blocks (see Section 2), managing distributed trust (see Section 2.4), and throughput management (see Section 2.5) are not onerous. Thus, the operational cost for running an OBM is significantly lower than Bitcoin. LSB comprises a large number of IoT users, devices, and SPs. Thus, by acting as OBM, the companies can advertise their product to a broad range of IoT users with small operational cost. The authors in [40] have demonstrated that the extensive uptake on IoT devices is having a significant impact on the advertising industry and the business model. Some entities, e.g., SPs or cloud storages, can apportion part of their existing infrastructure to serve as OBMs and do not need to install new equipment. This may be particularly attractive to service and cloud storage providers.

5.2. Blockchain auditability

In recent years, there have been some instances where data from IoT devices has been used as evidence in criminal cases [17]. For example, Fitbit data (steps walked) was used to contradict the claims made by a suspect about the movement of the victim prior to the crime [32]. These instances highlight the possibility of using IoT device data and records of interactions with these devices for auditing purposes. In blockchain, all transactions are stored permanently. Consequently, the history of transactions generated by a node can be audited by exploring the corresponding ledger of transactions of that particular node. Recall from Section 2.2 that each transaction maintains the ID of its previous transaction which makes it possible to trace through the entire ledger to verify prior actions. As an example, in smart home, it is vital for the home owner to know who has accessed his IoT devices or their data. By keeping a record of all these interactions, LSB affords an easy way for such users to perform an audit of their IoT devices.

Table 5
Comparing the complexity of different gossiping algorithms [24].

Algorithm	Time	Messages
CK [16]	$O(polylog(n))$	$O(npolylog(n))$
Trivial	$O(d+\delta)$	$O(n^2)$
EARS	$O(\frac{n}{n-f}log^2n(d+\delta))$	$Onlog^3n(d+\delta)$
TEARS	$O(d+\delta)$	$O(n^7log^2n)$

5.3. Complexity analysis

In this section, we analyze the complexity in the consensus and the network overhead. The complexity analysis demonstrates the rate with which the overheads grow as the network scales. The delay in consensus algorithm refers to the delay for the participating nodes to run the consensus algorithm and append new blocks to the blockchain. This delay is not affected by the scale of the network, thus, DTC functionality will not be impacted by the network scale. Recall from Section 2.5 that when the load in the network increases, the DTM adjusts consensus-period, which potentially impacts the delay in consensus, or the number of OBMs to manage the network throughput.

Given that the transactions are broadcast between the OBMs, the communication overhead complexity is inherited from the underlying peer-to-peer communication protocol. In [24] the authors analyzed the complexity of four algorithms which are shown in Table 5. The *Time* column in Table 5 represents the delay in communications and the "Messages" column represents the packet overhead.

In Table 5 n represents the number of processes, i.e., hops, d represents the communication delay, δ represents relative processing speed and f represents crash failure of each hop. Evidently, the time and message complexity are less in CK and EARS algorithms. Recall that LSB clusters the network to increase the scalability. This potentially significantly reduces n value in LSB compared with the studied methods, which in turn reduces the complexity of LSB. The complexity of the communication varies based on the underlying algorithm used in the peer-to-peer network.

6. Related works

In this section, we provide a literature review on IoT security and anonymity and blockchain-based systems.

IoT security: Authors in [42] proposed an end-to-end host identity protocol to secure IoT. The proposed method reduces the header size of the 6LowPAN and Host Identity protocol (HIP) from 40 bytes to a maximum of 25 bytes by eliminating unnecessary header fields and thus reduces network overhead. The authors also proposed a lightweight key distribution method for distributing keys between low resource IoT devices and users. A high resource available device is placed in the wireless range of the low resource devices to perform resource consuming tasks on behalf of the low resource devices. Although their approach is computationally lightweight for their considered particular application, removing the 6LowPAN and HIP header fields leads to reduced functionality. Moreover, the scalability of this approach

is limited due to the fact that the high resource device must be within wireless range of all IoT devices.

The authors in [33] proposed a new authentication and access control method to make IoT secure against unauthorized users and access. The proposed method relies on two authentication authorities namely: (i) Registration Authority (RA), and (ii) Home Registration Authority (HRA). The RA is designed to facilitate the authentication process for devices. All devices are registered with the RA. Similarly, the HRA facilitates the authentication process for the users. When a user wishes to access data from a particular device, the request is first sent to the RA. The RA checks the authenticity of the user with the HRA. Assuming the user is authenticated, the RA generates a shared key for communication between the user and the device. Security analysis shows that the proposed method is secure against the man-in-the-middle attack. However, the need for each device to have a RA and correspondingly each user to have a HRA could be a bottleneck for scalability. In LSB, we have rather proposed a structure where a single blockchain is managed distributedly by the overlay nodes and the devices authenticate other overlay nodes by updating keylists in OBM. Our approach scales better while also achieving protection against a broader range of attacks.

Blockchain applications: The notion of a blockchain was first introduced in the landmark paper [38] on Bitcoin by Satoshi Nakamoto. Bitcon aims to do away with centralized authorities for money exchange while offering a high level of security and privacy to the users. In 2013 a new blockchain platform, called Ethereum, was introduced [51]. Ethereum users are able to generate smart contracts with a small fee but with high security and privacy. Several applications have been proposed in recent years that make use of the Ethereum blockchain including blockchain in agriculture [3], crowd funding [5], and micro blogging [2].

Numerous other applications of blockchains have been proposed recently. Authors in [10] proposed a novel application of blockchain in energy trading. Using their proposed framework, energy producers can negotiate the selling price with their customers and also facilitate a smart contract to make a sale. A Distribution System Operator (DSO) ensures that the trade is secure and prevents the possibility for either a producer or customer to not follow through with their part of the contract. A lock key is used to prevent an energy producer from double spending (i.e. selling the energy to more than one customer). Security analysis shows that the framework is secure to a broad range of attacks. However, the architecture suffers from low scalability as a result of broadcasting all transactions and blocks to the whole network. In LSB, we overcome this challenge by limiting the number of nodes that manage the blockchain.

The authors in [27] proposed a blockchain-based multi-tier architecture to share data from IoT devices with organizations and people. The proposed architecture has three main components namely: data management protocol, data store system, and message service. The data management protocol provides a framework for data owner, requester, or data source to communicate with each other. The messaging system is used to increase the network scalability based on a publish/subscribe model. Finally, the data store system uses a blockchain for storing data privately. As in our work, they do not rely on POW given the associated overheads. In contrast to this work, we do not use the blockchain for storing user data as it will consume large bandwidth to store data in the distributed blockchain. Instead, we store hash of the data in the cloud in the blockchain.

The authors in [41] proposed a new ledger based cryptocurrency called IoTA. By eliminating the notion of blocks and mining, IoTA ensures that the transactions are free and verification is fast. The key innovation behind IoTA is the "tangle", which is essentially a directed acyclic graph (DAG). Before a user can send a transaction, he has to verify two randomly chosen transactions generated by other users. As the number of nodes increases, the transactions generated also increase but so do the number of transactions that are verified. LSB shares some similarities with IoTA such as zero transaction fees and both realize a self-scaling network. However, LSB employs a blockchain unlike the DAG employed by IoTA. LSB thus benefits from the inherent benefits of a blockchain such as the auditability offered by an immutable ledger.

The authors in [35] proposed a blockchain-based reputation system to establish trust in vehicular networks. The reputation of each vehicle is built progressively as the transactions generated by the vehicle are verified by other participating nodes. The transactions generated by more reputed nodes are accepted by the participating nodes. Similar to this work, LSB uses distributed trust. However, in LSB the trust is used to reduce the processing overhead for verifying new blocks which potentially significantly reduces the processing overhead in the blockchain.

In [7] the authors proposed GraphCoin, a scalable solution for digital financial transactions. GraphCoin uses PoS as the underlying consensus algorithm and therefore has the same limitations for IoT as PoS (see discussion in the rest of this section). To ensure the privacy of the users, GraphCoin employs Zerocoin technology where tracing the identity of the nodes is no longer possible. This increases the anonymity level for GraphCoin users.

In Catena [48], the authors proposed a method to reduce the overhead for auditing transactions in blockchain and increasing throughput. A server creates logs of a number of statements, i.e., off-the-chain communications between multiple parties. The logs, which are basically the hash of a number of statements, are then broadcast in the form of transactions to the Bitcoin blockchain. To audit blockchain transactions, the participants require to store the block headers and the Merkle tree of the block. The existence of a transaction can be proved using the Merkle tree. Catena increases the blockchain throughput as a single transaction stored in the blockchain contains the hash of a number of off-the-chain statements. However, Catena suffers from the following limitations: (i) as a single hash is stored in place of multiple statements, the blockchain participants cannot verify the content of a statement. The blockchain only will store the log of statements and is basically considered for logging purpose, (ii) mining delay of Bitcoin will still impact the stored transactions, as Bitcoin blockchain is used as the underlying blockchain solution, (iii) usage of Catena is limited to applications only where participants require to store a log in the blockchain and it may not be applicable for transactions with outputs, i.e., UTXO, or applications where participants need to verify the content of transactions, e.g., smart contracts, (iv) it relies on a central server to manage the logs.

In our recent work [22], we proposed a memory optimized and flexible blockchain known as MOF-BC. Blockchain is an append only database where removing transactions is not possible. However, in IoT with millions of users and thus millions of transactions, one can expect the size of the blockchain to increase significantly. This makes blockchain management challenging. MOF-BC enables the IoT users to remove their transactions from the blockchain while maintain the blockchain consistency. In MOF-BC, a user may store a single hash in place of multiple transactions to reduce the blockchain memory consumption. However, similar to Catena, the auditability and applicability of these transactions are limited.

Blockchain consensus algorithms: Ethereum is a blockchain instantiation that allows the participants to run distributed applications. Ethereum uses Proof of Stake (PoS) as the underlying consensus algorithm [51]. PoS requires miners to lock certain assets to be able to mine blocks. The mining power of each

Table 6
A summary of the discussed consensus algorithms.

Algorithm	Mining Power	Miner Anonymity	Packet Overhead	Processing Overhead	Throughput	IoT Limitation
PoS [3]	The amount of locked assets	Anonymous	Low	Medium	Low	- May lead to centralization - Requires the miners to pre-purchase asset
PoA [52]	The reputation	known	Low	Low	Medium	- Miners are known to the network - Mining is limited to pre-approved nodes
POET [53]	Time-based enforced by hardware	Anonymous	Low	Medium	Medium	- Requires the miners to be equipped with a particular hardware
AlgoRand [54]	Leader election based on voting	Anonymous	High	High	Low	- Low scalability due to voting
RepuCoin [55]	The reputation	Known	High	Medium	Low	- Miners are known to the network - Low scalability due to voting
FBA [56]	Leader election based on quorum intersections	Anonymous	High	Medium	Low	- Low scalability due to quorum formation and finding intersections
DTC	Time-based enforced by distributed neighbor monitoring	Anonymous	Low	Low	Self-scalable	- In small networks may be vulnerable to Sybil attack

node depends on the amount of locked assets. The miner with more locked assets has higher weight in mining blocks. PoS relies on the fact that the nodes that benefit most from the blockchain, i.e., have more assets invested in it, are unlikely to attack the blockchain (i.e., avoid self-harm). In IoT, large companies, e.g., Google, can purchase large portion of assets and thus eventually PoS may lead to centralization. Unlike PoS, the proposed DTC does not require miners to pre-purchase any asset in the blockchain, and thus all network participants can choose to be miner. DTC allows each OBM to store one block in each consensus-period.

Proof of Authority (PoA) is a consensus algorithm that can be considered as a form of PoS, wherein, the mining power of each miner is defined based on its identity in the network rather than the amount of locked assets [19]. A pre-approved group of nodes act as miner and their identity is known to all participants in the network. The miner with higher reputation have higher chance in mining new blocks. Unlike PoA, in DTC the miners are not limited to a pre-approved list of nodes. Any node can be elected to act as OBM and mine new blocks. Additionally, in DTC the PK of each OBM is not linked to its real-world identity thus providing a level of anonymity.

Recently Intel has designed a new consensus algorithm for blockchain known as Proof of Elapsed Time (POET) which is integrated with Hyperledger [12]. POET is a leader election algorithm which is intended to run in a Trusted Execution Environment (TEE) in Intel CPUs. Before a node can store a block in the blockchain, it must wait for a random time which is selected from a trusted enclave. A TimeChecker function verifies the choice of the random time. The block can only be appended to the blockchain after this time period. DTC (see Section 2.3) used in LSB is conceptually similar to POET. However, DTC does not rely on a particular hardware platform and is thus more generalized.

AlgoRand [26] is a consensus algorithm based on Byzantine agreement where the validators, i.e., miners, can reach agreement in one round. The validator of the next block is chosen randomly. The new block is propagated to the network and each validator votes to one block. Once all validators vote to the block, the agreement can be achieved. The block that has more votes is chosen as the next block. Similar to DTC, in AlgoRand there is no mining reward for validators. However, in IoT with large number of validators the number of blocks that are broadcast to the network significantly increases which in turn consumes significant bandwidth overhead from the participants. DTC does not require miners to vote on new blocks and thus incurs little packet overhead for achieving consensus.

The authors in [53] proposed a reputation-based consensus algorithm known as RepuCoin. In this algorithm, the reputation of the validators impacts their mining power, i.e., the more reputable nodes have a greater chance of adding new blocks to the blockchain. First, the highly reputable nodes create a group. The identity of the members of this group is known to the participants to measure reputation. A leader is chosen randomly among the members who is responsible for mining the next block. To choose the leader, each member of the group votes for one of the other members. The vote of each member is weighted using the reputation of the node. The voting increases the packet overhead particularly in IoT with large number of validators. The DTC algorithm does not require any voting between participants thus reduces the packet overhead as compared to RepuCoin. Unlike RepuCoin, the identity of the OBMs is not required to be known to other OBMs which in turns provides a level of anonymity.

In [36] the authors proposed Federated Byzantine Agreement (FBA). This algorithm is a distributed version of BFT where any node can choose to be validator. Each validator randomly selects a group of other validators to form a quorum. If the quorums have intersections, the nodes reach consensus on the leader who is tasked with mining the next block. In case that there are no intersections between the participants, the network might be disjoint. Each disjoint quorum independently decides on a leader to mine a new block, thus may potentially lead to inconsistency between blocks. The process of forming a quorum and checking for intersections incurs processing and packet overheads at the participating nodes. However, in DTC the OBMs are not required to form quorum.

One of the fundamental limitations of the existing consensus algorithms (as discussed above) is their limited throughput. To ensure the security of the blockchain, the existing solutions limit the number of blocks that can be appended to the blockchain which in turn limits the blockchain throughput. In the proposed DTC, the rate of block generation can be adjusted, based on DTM, while maintaining the blockchain security that in turn adjusts the network throughput. Thus, DTC allows blockchain to scale for larger networks as in IoT. Table 6 summarizes key properties of the consensus algorithms outlined in this section and presents a comparison with DTC.

7. Conclusion

In this paper, we argued that although Blockchain is an effective technology for providing security and anonymity in IoT, its application in the IoT context presents several significant challenges including: complexity, bandwidth and latency overheads and scalability. To address these challenges, we proposed a Lightweight Scalable Blockchain (LSB) for IoT. LSB has an IoT friendly consensus algorithm that eliminates the need for solving any puzzle prior to appending a block to the blockchain. LSB incorporates a distributed trust method whereby the processing time for validating new blocks by the OBMs gradually decreases as they build up trust in each other. A distributed throughput management strategy adjusts certain system parameters to ensure that the network utilization is within a prescribed operating range. Security analysis demonstrates that LSB is highly secure against a broad range of attacks. In the instance when key nodes fail, LSB operation exhibits graceful degradation, thus making it highly fault tolerant. Simulation results show that the proposed architecture decreases bandwidth and processing time compared to the classical blockchains. Generally, LSB brings a high level of security and anonymity for IoT users while enforcing a marginal overhead.

In our future work, we plan to develop a prototype implementation of LSB to understand its performance in real-world settings. We will also explore the suitability of LSB in other application domains such as smart grids and vehicular networks.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.08.005.

References

[1] Open Shortest Path First(OSPF), R.F.C. 2328. https://www.ietf.org/rfc/rfc2328.txt. (Online; Accessed 19 November 2016).
[2] Eth-twitter, 2017, www.github.com/yep/eth-tweet. (Online; Accessed July 2017).
[3] Fullprofile, 2017, www.fullprofile.com.au. (Online; Accessed July 2017).
[4] NS3, 2017, https://www.nsnam.org. (Online; Accessed July 2017).
[5] Wei-fund, 2017, www.weifund.io. (Online; Accessed July 2017).
[6] Blockchain explorer, 2019, https://www.blockchain.com/charts/hash-rate?timespan=1year.
[7] Graph coin, 2019, https://graphcoin.net/graphcoinwhitepaper.pdf.
[8] Miner gate, 2019, https://minergate.com/.
[9] M. Abramaowicz, Cryptocurrency-based law, Ariz. L. Rev. 58 (2016) 359.
[10] N.Z. Aitzhan, D. Svetinovic, Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams, IEEE Trans. Dependable Secure Comput. (2016).
[11] Amazon, Amazon echo, 2018, https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/.
[12] A. Baliga, Understanding Blockchain Consensus Models, Tech. Rep., Persistent Systems Ltd, 2017.
[13] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, I. Verbauwhede, Spongent: A lightweight hash function, in: B. Preneel, T. Takagi (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 312–325, http://dx.doi.org/10.1007/978-3-642-23951-9_21.
[14] C. Cachin, Architecture of the Hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.
[15] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, W. Shi, On security analysis of proof-of-elapsed-time (poet), in: International Symposium on Stabilization, Safety, and Security of Distributed Systems, Springer, 2017, pp. 282–297.
[16] B.S. Chlebus, D.R. Kowalski, Time and communication efficient consensus for crash failures, in: International Symposium on Distributed Computing, Springer, 2006, pp. 314–328.
[17] CNN, Suspect OKs Amazon to hand over Echo recordings in murder case, 2018, https://edition.cnn.com/2017/03/07/tech/amazon-echo-alexa-bentonville-arkansas-murder-case/index.html.
[18] D. Cooper, Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile, 2008.
[19] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, V. Sassone, PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain, 2018.
[20] V. Dedeoglu, R. Jurdak, G.D. Putra, A. Dorri, S.S. Kanhere, A trust architecture for blockchain in IoT, 2019, arXiv:1906.11461.
[21] H. Delfs, H. Knebl, H. Knebl, Introduction to cryptography, Vol. 2, Springer, 2002.
[22] A. Dorri, S.S. Kanhere, R. Jurdak, MOF-BC: A memory optimized and flexible blockchain for large scale networks, Future Gener. Comput. Syst. 92 (2019) 357–373.
[23] A. Dorri, C. Roulin, R. Jurdak, S. Kanhere, On the activity privacy of blockchain for IoT, 2018, The 44th IEEE Conference on Local Computer Networks (in press).
[24] C. Georgiou, S. Gilbert, R. Guerraoui, D.R. Kowalski, On the complexity of asynchronous gossip, in: Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing, ACM, 2008, pp. 135–144.
[25] A. Gervais, G.O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 3–16.
[26] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, ACM, 2017, pp. 51–68.
[27] S.H. Hashemi, F. Faghri, P. Rausch, R.H. Campbell, World of empowered IoT users, in: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), IEEE, 2016, pp. 13–24.
[28] S. Huh, S. Cho, S. Kim, Managing IoT devices using blockchain platform, in: Advanced Communication Technology (ICACT), 2017 19th International Conference on, IEEE, 2017, pp. 464–467.
[29] S.J. Koh, M.J. Chang, M. Lee, MSCTP for soft handover in transport layer, IEEE Commun. Lett. 8 (3) (2004) 189–191.
[30] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: Security and Privacy (SP), 2016 IEEE Symposium on, IEEE, 2016, pp. 839–858.
[31] A. Kousaridas, S. Falangitis, P. Magdalinos, N. Alonistioti, M. Dillinger, SYSTAS: Density-based algorithm for clusters discovery in wireless networks, in: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on, IEEE, 2015, pp. 2126–2131.
[32] Lifewire, IoT device witness, 2018, https://www.lifewire.com/when-tech-is-witness-to-murder-4149689.
[33] J. Liu, Y. Xiao, C.L.P. Chen, Authentication and access control in the internet of things, in: 32nd International Conference on Distributed Computing Systems Workshops, 2012, pp. 588–592 http://dx.doi.org/10.1109/ICDCSW.2012.23.
[34] K. Lu, Y. Qian, M. Guizani, H.-H. Chen, A framework for a distributed key management scheme in heterogeneous wireless sensor networks, IEEE Trans. Wireless Commun. 7 (2) (2008).
[35] Z. Lu, Q. Wang, G. Qu, Z. Liu, Bars: a blockchain-based anonymous reputation system for trust management in vanets, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2018, pp. 98–103.
[36] D. Mazieres, The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus, Stellar Development Foundation, Citeseer, 2015.
[37] K. Mikhaylov, J. Petaejaejaervi, T. Haenninen, Analysis of capacity and scalability of the LoRa low power wide area network technology, in: European Wireless 2016; 22th European Wireless Conference; Proceedings of, VDE, 2016, pp. 1–6.
[38] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008.
[39] D. Ongaro, J.K. Ousterhout, In search of an understandable consensus algorithm, in: USENIX Annual Technical Conference, 2014, pp. 305–319.
[40] O. Petrovic, 3.3 the internet of things as disruptive innovation for the advertising ecosystem, 2017.
[41] S. Popov, The tangle, 2016, p. 131, cit. on.
[42] S. Sahraoui, A. Bilami, Compressed and distributed host identity protocol for end-to-end security in the IoT, in: 2014 International Conference on Next Generation Networks and Services, NGNS, 2014, pp. 295–301. http://dx.doi.org/10.1109/NGNS.2014.6990267.
[43] F.-S. sense, www.sense.f-secure.com. (Online; Accessed November 2016).
[44] P.K. Sharma, S. Singh, Y.-S. Jeong, J.H. Park, DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks, IEEE Commun. Mag. 55 (9) (2017) 78–85.
[45] A. Sivanathan, D. Sherratt, H.H. Gharakheili, V. Sivaraman, A. Vishwanath, Low-cost flow-based security solutions for smart-home IoT devices, in: Advanced Networks and Telecommunications Systems (ANTS), 2016 IEEE International Conference on, IEEE, 2016, pp. 1–6.

[46] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, ACM SIGCOMM Comput.Commun. Rev. 31 (4) (2001) 149–160.
[47] T.S. Telecommunications, I. converged Services, P. for Advanced Networking (TISPAN), https://www.etsi.org/.
[48] A. Tomescu, S. Devadas, Catena: Efficient non-equivocation via Bitcoin, in: 2017 38th IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 393–409.
[49] F. Tschorsch, B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies, IEEE Commun. Surv. Tutor. 18 (3) (2015) 2084–2123.
[50] M. Vukolić, The quest for scalable blockchain fabric: Proof-of-work vs. Bft replication, in: International Workshop on Open Problems in Network Security, Springer, 2015, pp. 112–125.
[51] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper, 151, 2014.
[52] L. Xu, S. Chainan, H. Takizawa, H. Kobayashi, Resisting sybil attack by social network and network clustering, in: Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on, IEEE, 2010, pp. 15–21.
[53] J. Yu, D. Kozhaya, J. Decouchant, P. Esteves-Verissimo, RepuCoin: Your reputation is your power.
[54] X. Yue, H. Wang, D. Jin, M. Li, W. Jiang, Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control, J. Med. Syst. 40 (10) (2016) 218.
[55] ZDNet, Zdnet, 2017, http://www.zdnet.com/article/a-better-blockchain-bitcoin-for-nothing-and-transactions-for-free/.
[56] 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: Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on, IEEE, 2014, pp. 230–234.

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 research interest includes blockchain applications and challenges in adopting blockchian in large scale networks including the Internet of Things, smart cities, smart grid, and e-health. He has published over 20 peer-reviewed papers.

Salil S. Kanhere received his M.S. and Ph.D. degrees, both in Electrical Engineering from Drexel University, Philadelphia, USA. He is currently a Professor in the School of Computer Science and Engineering at the University of New South Wales in Sydney, Australia. His research interests include Internet of Things, blockchain, applied machine learning and cybersecurity. He has published over 200 peer-reviewed articles and delivered over 35 tutorials and keynote talks on these research topics. He is a contributing research staff at Data61, CSIRO and the Cybersecurity Cooperative Research Centre. He has held visiting positions at TU Darmstadt, TU Graz, University of Zurich and the Institute for Infocomm Research, Singapore. Salil regularly serves on the organizing committee of a number of IEEE and ACM international conferences including IEEE PerCom, ACM MobiSys, ACM SenSys, ACM CoNext, IEEE WoWMoM, IEEE ICBC and IEEE LCN. He currently serves as the Editor in Chief of Ad Hoc Networks and on the Editorial Board of Pervasive and Mobile Computing, and Computer Communications. Salil is a Senior Member of both the IEEE and the ACM and an ACM Distinguished Speaker. He is a recipient of the Humboldt Research Fellowship.

Raja Jurdak received the MSc in electrical and computer engineering, and the PhD degree in information and computer science, both from the University of California, Irvine. He is a Professor of Distributed Systems and Chair in Applied Data Sciences at Queensland University of Technology. 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 150 peer-reviewed publications, including a sole-authored book titled: "Wireless ad hoc and sensor networks : a cross-layer design perspective". He serves on the editorial board of 4 international journals, including Ad Hoc Networks, and on the organising and technical program committees of top international conferences, including Percom, ICBC, and IPSN. He is an honorary professor with the University of Queensland, conjoint professor with the University of New South Wales, and a senior member of the IEEE.

Praveen Gauravaram is a Senior Scientist and Innovation Evangelist at Tata Consultancy Services Limited (TCSL), Australia and an Adjunct Senior Lecturer in the School of Computer Science and Engineering at UNSW, Australia. Praveen's focus is on embedding innovation & creativity into TCS's customer deliverables and offerings in Australia & New Zealand (ANZ) geography. While his primary focus is on cyber security, Praveen leads TCS's Research & Innovation activities in other research areas and programs that add immense value to TCS's business. Praveen has a PhD in Cryptology from Queensland University of Technology, Brisbane, Australia. Praveen has held scientific positions in India, Europe and Australia and is a recipient of research grants and awards whist his postdoctoral fellowship at Technical University of Denmark. Praveen's significant scientific achievements include co-design of Grøstl cryptographic hash function, a finalist in the SHA3 cryptographic hash competition conducted by US National Institute of Standards and Technology and security evaluation of standard cryptographic designs. To date, Praveen has published more than fifty research papers in several top conferences, journals and research consortiums and has co-authored articles with at least fifty researchers from several geographies including Asia, Asia-Pacific, Europe, and North America.