Blockchain of Finite-Lifetime Blocks With Applications to Edge-Based IoT

Chan Kyu Pyoung and Seung Jun Baek, *Member, IEEE*

***Abstract*—Edge computing is a promising approach for provisioning distributed cloud services to Internet of Things (IoT) systems. Many recent studies propose that edge nodes use blockchain for the decentralized management and access control of IoT data. However, due to the massive volume of data and related transactions, edge servers will eventually run out of space to store the full chain. We introduce scalable and lightweight architecture called LiTiChain, a blockchain of blocks with finite lifetime. In LiTiChain, outdated transactions and blocks, that is, the blocks whose lifetimes are expired, can be safely removed from the chain. Two graphs are merged into the structure of LiTiChain: 1) a tree representing the order of expiry of lifetimes and 2) a linear graph representing the order of block creation. We show that this construction not only ensures the connectivity of the chain after block deletions but also helps to maintain the block height of shortened chain. LiTiChain also supports transactions whose lifetime is unknown at the time of creation. It is possible that some expired blocks need to be retained in the chain, in case they are needed to validate remaining blocks, which incurs additional storage costs. A detailed analysis of such overhead in storage costs is presented for stochastic and worst case scenarios. Extensive simulation is performed on actual and synthetic IoT data so as to gain insights on the storage costs under various lifetime distributions. It is demonstrated that LiTiChain provides a simple yet effective solution to scalability problems in storing blockchains for the IoT ecosystems.**

***Index Terms*—Blockchain, edge computing, Internet of Things (IoT), security, storage costs.**

I. Introduction

THE INTERNET of Things (IoT) industry continues to grow fast, and it is projected that over 60 billion devices will have the Internet connectivity by 2025 with the global market size surpassing a trillion dollars [1]. As IoT becomes prevalent in daily life handling sensitive and private data, the IoT security has become a major concern. With an explosive number of connected devices, it is challenging to address issues in privacy, device reliability, hacking, and data integrity. In particular, centralized approaches to the IoT security have faced limitations due to the massive scale of generated data. To overcome these problems, a number of recent studies took decentralized approaches using blockchain [2]–[5].

Manuscript received July 25, 2019; revised October 22, 2019 and November 5, 2019; accepted December 3, 2019. Date of publication December 13, 2019; date of current version March 12, 2020. This work was supported by the National Research Foundation of Korea grant funded by the Korea Government (MSIT) under Grant 2018R1A2B6007130. *(Corresponding author: Seung Jun Baek.)*

The authors are with the Computer Science and Engineering Department, Korea University, Seoul 02841, South Korea (e-mail: pyoung1101@korea.ac.kr; sjbaek@korea.ac.kr).

Digital Object Identifier 10.1109/JIOT.2019.2959599

Blockchain is a peer-to-peer distributed ledger which has initially gained success with cryptocurrencies such as Bitcoin [6]. Because blockchain is nearly forge- and tamper-proof, it can handle secure transactions and data processing without authoritative intermediary. Blockchain is also characterized by security, transparency, integrity, and decentralization and is currently applied to various fields, such as finance, manufacturing, health care, etc., [7]. We consider a scalable blockchain architecture suitable for IoT in the edge computing environment.

Due to the sheer number of connected IoT devices, centralized cloud architecture has scalability problems in gathering and processing generated data. The edge computing framework [8]–[10] alleviates the problems by bringing computation, data processing, and storage services closer to the IoT devices. In edge computing, the first-hand processing of IoT data is done at the edge servers which may compress and summarize the collected data. The summarized data in a greatly reduced size can be later forwarded to the central cloud. The edge layer can offload not only data processing tasks from cloud but also low-latency compute-intensive tasks from the IoT devices. Recently, there has been much interest in machine-learning tasks which are delay-sensitive requiring quick inference for speech recognition, translation, image classification, and video recognition [11]. Edge computing can thus effectively reduce power consumption and computational load of IoT devices [12]–[17].

We envision edge-based IoT systems where the edge servers form a distributed network supporting storage and sharing of IoT data using blockchain. The idea has been explored in prior works; for example, EdgeChain [18] is proposed to record the activities and transactions among IoT along with the resource management. Also, a model for sharing economy services based on blockchain is proposed [19], where the intelligent processing of multimedia and cyber-physical data occurs at edge nodes which handle key transactions through blockchain. A decentralized storage system for IoT data with blockchain-based access control was proposed [20] on top of the decentralized cloud architectures such as Cloudlets [21]. The challenge with managing IoT data through blockchain is, however, the scalability in the storage capacity. Currently, for Bitcoin, more than 225 GB is required to store the full chain. The transaction rate of Bitcoin is typically less than ten per second. By contrast, the generation rate of data from massive-scale IoT devices and the associated transactions will be significantly higher than those of monetary transactions in cryptocurrencies. Because each edge server has a direct control

over the associated client IoT devices and their data, each server should be able to fully verify transactions and blocks, i.e., should be a "full node," and thus should store the full chain. The blockchain is, by design, an ever-growing, permanent record of data blocks. Thus, the scalability of storage is problematic with existing approaches, considering that edge servers typically have limited storage space.

In this article, we introduce a scalable and lightweight blockchain architecture called LiTiChain (pronounced as *litty*[1]-chain). LiTiChain is a blockchain of blocks with finite lifetime. If the lifetime of a block expires, the block can be *deleted* from the chain. Thus, the size of the chain does not necessarily grow monotonically over time. The lifetime of a block is determined by the lifetime of transactions within the block; that is, we mainly consider transactions with lifetimes.

In addition to storage issues, we give rationale behind considering IoT data and transactions with lifetimes. IoT generates time-series data, such as sensor measurements, event logs, environmental and behavioral data, etc. Certain types of transactions may lose value over time. For example, suppose a customer equipped with an IoT device gets issued with a coupon with expiration date. The issuer of the coupon needs to keep track of the transaction (issuance) until the expiration date; however, after the expiration date, there is little point in storing that information. The *depreciation* of data and its transactions over time is common in IoT applications.

Also, there is a growing concern in the privacy of blockchains. Blockchain is pseudonymous, where a participant is identified by its public key or hash. However, it is possible to infer the actual identity of users by analyzing their transaction patterns recorded in the ledger [22], [23]. This issue can be partly addressed if some outdated transactions can be erased in a timely manner. Besides, the IoT data shared among participants may contain personal information which its owner may not want to disclose permanently. Thus, a user may request data or transaction history to be removed from the record. Blockchain has evolved into a platform not only for cryptocurrency but also for transactions in healthcare and insurance which contain sensitive personal information, such as disease and genetic information. General data protection regulation (GDPR) [24] in European Union (EU) protects the fundamental rights to the privacy of data subjects in Europe. The right to be forgotten (Article 17) stipulates that the record shall be removed if the purpose of collecting personal information has been fulfilled or that the information subject has withdrawn its consent. Thus, personal information should be considered for deletion if it is not necessary for the purpose of use or if the retention period has passed. If a user intentionally uploads data with explicit contents or private information without consent, the data will permanently remain in the blockchain. Thus, it is desirable to remove data that conflicts with societal norms and values from the storage even without agreements with the uploader.

The lifetime of a transaction, however, may not be always available when the transaction is created. LiTiChain is designed to handle three types of transactions: those with a *fixed*, *permanent*, or *indefinite* lifetime depending on if the lifetime is either deterministically finite, infinite, or undetermined, respectively, at the time of creation. This aspect complements the traditional blockchain which is not intended for handling data of which the validity lasts for a finite duration.

We propose and explore the architecture of LiTiChain in the rest of this article. Due to the variability of block lifetimes, the topology of the proposed blockchain may change over time. Particularly, if past blocks are deleted, the blochchain data structure may become disconnected. We thus propose a blockchain structure which merges two types of graphs: the first graph with a tree structure based on the order of expiration of lifetimes; and the second graph with a linear structure based on the order of creation as in the conventional chain. We show that the former structure ensures the connectivity of the chain, whereas the latter increases the height of the chain, i.e., the distance to the genesis block, leading to an improved strength of security of the chain.

Meanwhile, a block may be retained in LiTiChain although its lifetime has expired because there may exist other blocks which refer to the block for validation purposes. The deletion of the referred block will be delayed, which incurs additional storage costs. We will discuss methods to reduce such costs due to overdue blocks. A detailed analysis on how the retention overhead will affect the total storage cost is provided as well. We summarize our contributions as follows.

1) We propose a scalable blockchain architecture called LiTiChain, which manages transactions with finite lifetimes. In LiTiChain, the expired blocks can be removed from the chain while maintaining its connectivity. LiTiChain provides a simple solution to storage problems of conventional blockchains and is suitable for edge-based IoT ecosystems generating a large number of transactions.
2) We present both stochastic and worst case analysis on the storage costs incurred by retained blocks. For the stochastic model, we derive the expected cost of retention overhead in a closed form. We also obtain an upper bound on the worst-case retention cost (RC) of overdue blocks.
3) We extensively perform numerical experiments using the actual and synthetic IoT data sets. We examine and provide insights on the storage costs, topology changes, and retention overhead of LiTiChain.

The remainder of this article is organized as follows. We review the related work in Section II. We describe the system model of our proposed blockchain in Section III. The algorithms for block creation and deletion are presented in Section IV. In Section V, we analyze the worst case in our proposed blockchain. The performance evaluation via simulations is presented in Section VI. Section VII concludes this article.

II. Related Work

The conventional centralized approaches to IoT security can be vulnerable in large scale systems. To address the scalability issues, decentralized methods based on blockchain are recently

[1]Although now obsolete, *litty* meant *little* in the English literature.

proposed. Dorri *et al.* [25] proposed a decentralized, privacy preserving, and secure architecture based on blockchain for interconnected smart vehicles. The privacy of user was ensured by changing the public key for each transaction. The authors found various applications of their architecture to the automotive industry, such as remote software updates, insurance, smart charging for electric vehicles, and car-sharing services. Novo [26] introduced a distributed access control system for IoT using blockchain. The IoT devices are indirectly connected to blockchain through management hub nodes, where the hub node used a signed certificate allowing the IoT devices to verify its authenticity. Wan *et al.* [27] proposed a partially decentralized Industrial IoT (IIoT) architecture for smart factory based on blockchain. They combined the Bell-La Padula model and the Biba model to ensure confidentiality, integrity, and availability of data and service. Their proposed scheme achieved an improved security and privacy protection than the centralized IIoT architecture in automatic production platforms.

Next, we introduce scalable systems and solutions for blockchain. InterPlanetary file system (IPFS) [28], [29] is a decentralized and distributed file system with high integrity and resiliency. Unlike HTTP Web, IPFS maintains a stable system because data can be shared by connected nodes even in the presence of disconnected nodes. In order to maintain the network, miners are paid with Filecoin [30] as a reward for providing data storage and retrieval. Sharding [31] is an on-chain solution that divides the entire network into *shards* in order to store transactions separately, and then to process the stored transactions in parallel. Plasma [32] is an off-chain solution that enables scalable transactions on Ethereum by introducing blockchain-in-blockchain structure consisting of child- and parent-chain. Transactions are offloaded to and are processed by the child-chain, where the processed transactions are verified and stored at the parent chain in the form of the Merkel root of their hashes.

The following studies consider an architecture combining IoT, blockchain, and edge computing. Sharma *et al.* [33] presented a distributed cloud architecture using blockchain. In the edge of the IoT network, a distributed fog node consisting of software-defined networking (SDN) controllers based on blockchain is used for a low-latency service of computing resources. Fog nodes are used as assisted computing resources where the raw data is stored in distributed cloud storage. Pan *et al.* [18] designed an edge-IoT framework called "EdgeChain" based on blockchain and smart contracts. EdgeChain is a credit-based resource management system to control the resources offered to IoT devices by the edge servers. Kang *et al.* [34] proposed a reputation-based sharing of vehicle data with consortium blockchain and smart contract. Their scheme is intended for secure and efficient data management in the vehicular edge network, where the control and storage of data is done at the edge nodes which are the roadside units (RSUs).

The scalability is crucial for the storage of a massive amount of data collected from IoT devices. Blockchain has been applied to the storage on clouds [35], [36] as well as the distributed storage [37], [38]. In the following, we summarize works on distributed storage for IoT data empowered by blockchain. Liang *et al.* [39] proposed DroneChain to ensure the secure communication and integrity of data collected from drones. They simulated the average response time of data transmission with an increasing number of drones and showed the effectiveness in scalability. Biswas *et al.* [40] proposed a scalable blockchain framework for IoT, where they created a local peer network and achieved increased transaction rate and the ledger scalability through all peers. Jiang *et al.* [41] integrated Internet of Vehicles (IoV) and blockchain for distributed and secure storage of big data. Li *et al.* [42] proposed the blockchain-based large-scale IoT data storage and protection. The features of the proposed distributed storage are: the IoT data are stored off-chain; the access to IoT data is controlled by the majority of miners; and all access activities are recorded in the blockchain. Xu *et al.* [43] introduced Healthchain, a privacy-preserving system for large-scale health data based on blockchain, where the IoT data and doctors' diagnoses are stored in IPFS.

The studies in [44] and [45] consider redactable blockchains at the block- and transaction-level, respectively. Deuber *et al.* [44] proposed a redactable blockchain that avoids heavy cryptographic primitives and used a consensus-based voting in the permissionless setting. The proposed chain consists of blocks which are linearly connected by two links, the old and the new link. If a new block in the candidate block pool receives approval votes in the majority, the old block can be replaced with the new one. The old state of the block is maintained for validation purposes. However, the chain length and structure remain unchanged under their framework, and the work does not address the storage problem for blockchains. Derler *et al.* [45] introduced a policy-based chameleon-hash functions (PCHs) for fine-grained and controlled rewriting in blockchain. The scheme uses the property of chameleon-hash functions such that the hash collisions can be generated after making arbitrary changes to blocks, only by authorities in possession of a secret key, which is otherwise collision resistant. The secret key which can "unlock" the interlocked chain of blocks (called the trapdoor key), which makes blocks modifiable, needs to be shared among the authorities. However, there is a risk of a small number of authorities entitled to redact transactions being compromised, in which case the entire system can fail.

None of the aforementioned works have explicitly addressed the issue of storing ever-growing blockchain along with the possibility of block removals. Conversely, LiTiChain can be applied to all of the above works, whenever there is need for lightweight blockchain which can delete outdated blocks so as to secure storage spaces for edge servers acting as full nodes.

III. Proposed Method

A. System Model

We consider an IoT system consisting of a IoT devices and a network of edge servers. Each IoT device is associated with an edge server in the vicinity. Each edge server is responsible for processing and storing data collected from a group of IoT devices, where the edge servers communicate with one another

in a peer-to-peer (P2P) manner through the Internet. The edge nodes may be associated with a single central cloud or multiple cloud service entities.

IoT devices generate data over time. Any activities on data, such as access, computation, storage, networking, etc., are regarded as *transactions*. The edge servers will use blockchain for secure record-keeping and control of transactions of IoT data. We assume that a permissioned blockchain is used such that only edge servers with permissions can participate in the blockchain network as a full node. Permissioned chain is also appropriate for processing transactions with a high throughput. Because of the lightweight and resource-constrained nature of IoT devices, the devices only generate and report raw data to edge nodes. Any activities, such as maintenance, storage, and computation related to the blockchain, are done by edge servers. Because a permissioned blockchain is adopted, edge servers may use a practical Byzantine fault-tolerance (PBFT) algorithm [46] to reach a distributed consensus. We assume that the block creation is validated through PBFT. PBFT may also be used for the block deletion. In case there exist adversaries which refuse to accept the deletion, the block is not deleted and remains in the blockchain until it is validated by PBFT. Once the validity of a block is verified, the verification is notified to other edge servers, and a new block is connected to the blockchain of each edge server. This is similar to verifying the validity of the Bitcoin blockchain.

The edge server has relatively limited storage compared to, e.g., central cloud servers. Thus, permanently storing the full history of IoT transactions can be problematic, especially, if a large number of IoT devices frequently generate data. Moreover, the value of event data collected by IoT devices may depreciate over time. To address scalability issues, we introduce LiTiChain where outdated blocks can be deleted as follows. Each transaction is assumed to have a finite *lifetime* after which the transactions losses value, and thus can be deleted from record. A block may contain multiple transactions with different lifetimes. We define the lifetime of a block as the time from the creation of the block to the latest *endtime* among transactions, where the endtime of a transaction/block as the absolute time index at which its lifetime expires. If the lifetime of a block expires, it can be deleted from the blockchain to secure storage resource at edge servers.

B. Ensuring Chain Connectivity: Endtime Ordering Graph

In conventional blockchain with linearly connected blocks, if a block is deleted before the endtime of a block which was created later, the blockchain will be disconnected. We consider structures which ensure the connectivity of blocks as follows. LiTiChain has a graph structure based on the order of *endtime* of the blocks. When a block is created, we create a directed edge emanating from the new block to an existing block. The edge direction represents that the hash of the new block is a function of the hash of the connected block. Below we outline how to insert a new block to the existing blockchain. The block created for the ith time is denoted by b_i for $i = 1, 2, \ldots$ The timestamp at which the ith block is created is denoted by t_i, and the *endtime* of b_i is denoted by e_i.

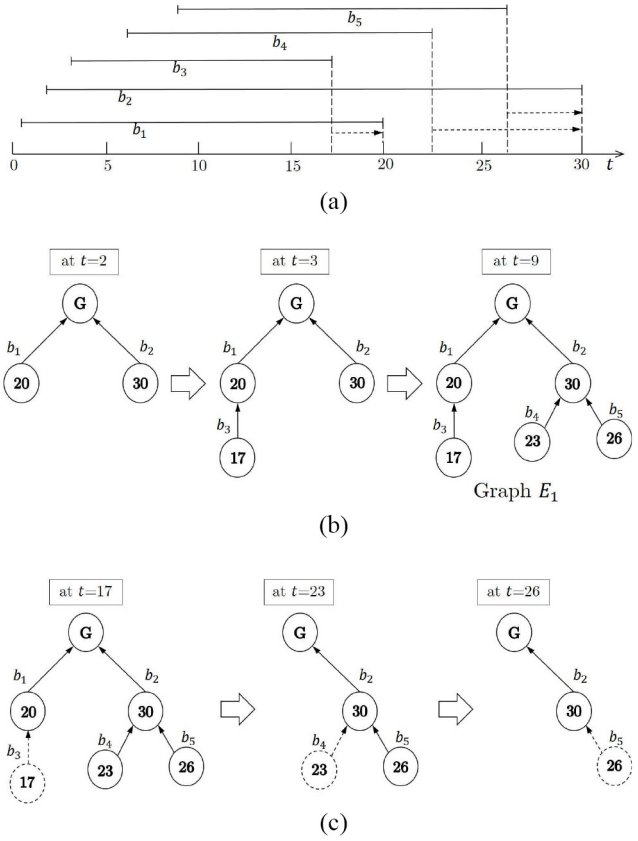

Fig. 1. Construction of EOG in LiTiChain. (a) Block lifetimes and endtimes. (b) Inserting blocks. (c) Deleting blocks.

1) Suppose there exist blocks whose endtime is later than that of the new block b_i. Among those blocks, b_i is connected to one with the earliest *endtime* by a directed edge from b_i to the connected block.
2) If all of the endtimes of existing blocks are earlier than that of the new block b_i, connect b_i to the genesis block G with a directed edge from b_i to G. The lifetime of G is assumed to be infinite.

The directed graph obtained from the above insertion rule is called endtime ordering graph (EOG). EOG will have a tree topology according to the precedence relation of endtimes, where the root node is the genesis block G. For a given directed edge, we will call the head of the edge as the parent node, and the tail of the edge as the child node. For block b_i, the parent node of b_i according to endtime ordering is denoted by b_i^* throughout this article. Fig. 1 shows an example of constructing EOG. The endtime e_i's is the blocks is shown in Fig. 1(a). The insertion of blocks is shown in Fig. 1(b), where the numbers in the block represents endtime e_i. When the lifetime of a block expires, the block is deleted from the blockchain as shown in Fig. 1(c). We observe that EOG is a connected graph at all times because a child node in the tree-structured EOG will always have earlier endtime than that of its parent node.

C. Increasing Block Height

A potential problem with EOG structure is that the height of a block, measured as the distance from the block to the

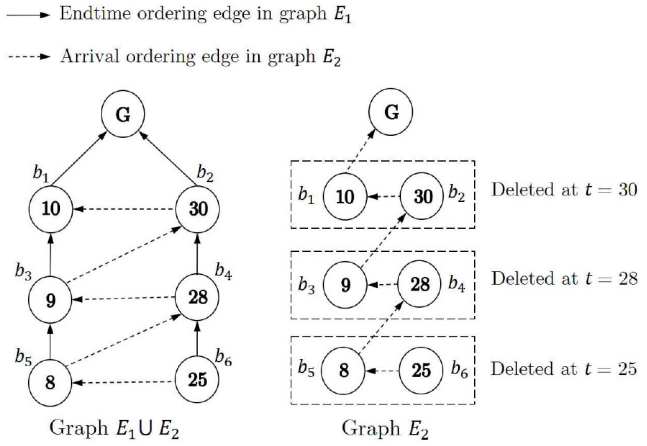

Fig. 2. Final graph of blocks is an union of EOG and AOG.

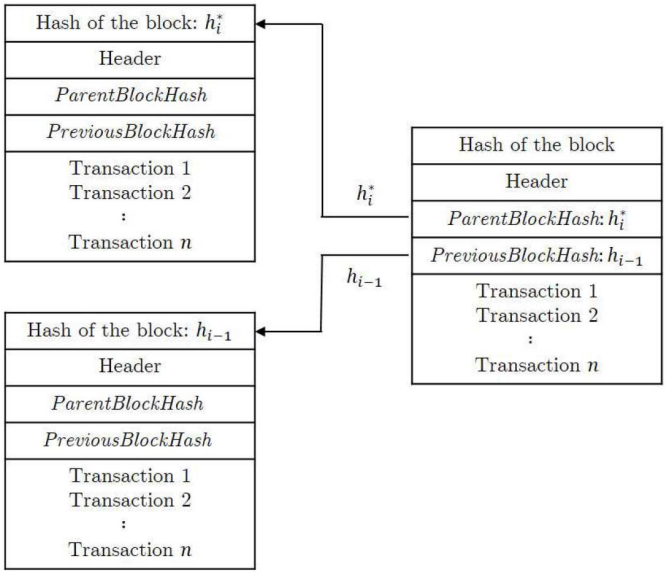

Fig. 3. Hash fields in the header of block b_i.

genesis, can be shallow. In other words, the length of the path starting from a block and following the edge directions tracking back to G, can be short. For example, in the rightmost case, the height of b_5 is only 2, although the total number of blocks excluding G is 5. Thus, in conventional blockchain, the height of b_5 would have been 5. Moreover, unlike conventional blockchains, the total size of blockchain may not always increase due to the deletion of blocks, which may create shallow branches. Longer chains are preferable from the security perspective, as can be seen from the so-called "longest chain rule" in Bitcoin blockchain [6]. When multiple new chains are created in the Bitcoin blockchain, the longest chain (in terms of the total difficulty) from the genesis block is chosen as the valid chain by miners. The rough idea is that, the longer the chain, it would be harder to undo that chain, thus the chain is considered to be more secure.

In order to address that issue, we consider a simple method to increase the block height of the EOG-based chain. We consider another graph of linear topology called the arrival ordering graph (AOG). AOG is simply a linear graph according to the order of arrival (creation) of blocks. Upon creation of a new block, say b_i, we form a directed edge from b_i to the previously and most recently created block in the chain which is denoted by b_i^-. For example, if b_{i-1} exists in the chain and was not deleted when b_i is created, then we have $b_i^- = b_{i-1}$. AOG has the linear linked-list structure of the conventional blockchain. LiTiChain is obtained by merging EOG and AOG; specifically, we take a union of the sets of edges of AOG and EOG. Thus, for block b_i, there will be two directed edges emanating from b_i, one to b_i^* and the other to b_i^-, determined from EOG and AOG, respectively. If $b_i^* = b_i^-$, b_i has one parent, and there will be only one edge from b_i to b_i^*. The height of a block is now defined as the *maximum* distance path on the edge union starting from the block to G. Note that the height of a block is also equivalent to the length of the path to G in AOG which is same as the *height* in Bitcoin blockchain. An example is shown in Fig. 2. On the right of Fig. 2 is the AOG, and on the left shows the final form of LiTiChain obtained from the union of EOG and AOG. The height of b_4 is 2 under EOG only; however, the height increases to 4 after merging.

Block Hash: The block header of conventional blockchain contains the hash of the previous block. By contrast, in LiTiChain, there are two fields in the block header which contain block hashes. For block b_i, we refer to b_i^* as *ParentBlock* and b_i^- as *PreviousBlock*. We also define two fields for hashes in the header of block b_i as follows.

1) `ParentBlockHash`: Hash of *ParentBlock* b_i^* of b_i.
2) `PreviousBlockHash`: Hash of *PreviousBlock* b_i^- of b_i.

Thus, the first hash implements the directed edge of EOG, and the second hash implements the directed edge of AOG. Fig. 3 shows two fields for hashes in the header of block b_i.

D. Retention Cost and K-Height Insertion

Suppose the endtime of a block, say b_i, has passed. Then b_i is an expired block and should be deleted from the chain. However, suppose there was block b_{i+1} which arrived next to b_i and its endtime e_{i+1} was later than e_i. Because b_{i+1} is not expired yet, b_i should not be deleted from the blockchain because b_i is necessary for verifying the validity of b_{i+1}. In this case, our rule is that b_i must remain in the chain until the expiry of b_{i+1}. This incurs extra storage cost because b_i must be retained beyond its endtime, and its expiry time is effectively extended. The storage cost of data is defined to be the product of the data size and its storage time. RC of a block is defined to be the additional storage cost incurred by the extended expiration time of the retained block.

The retention mechanism maintains the "hash-chained" property of the blockchain. This is because the validity of an arbitrary block and the blocks preceding it can be recursively verified for validation by the associated `ParentBlockHash` and `PreviousBlockHash`, because both *ParentBlock* and *PreviousBlock* of a block to be verified always exist in the chain during the block's lifetime (the former precedes in lifetime, and the latter is retained) and are available for validation. Note that `PreviousBlockHash` field in the header through

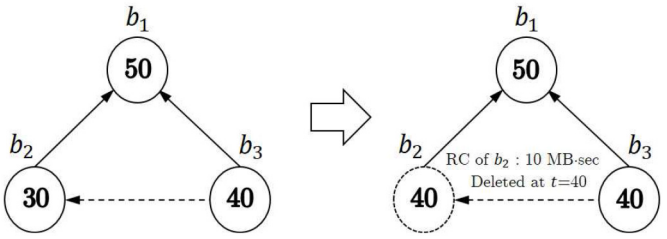

Fig. 4. RC incurred by the extended expiration time of a block.

which the block validity can be verified is similar to the field of *previous block hash* in Bitcoin blockchain.

An example is shown in Fig. 4. The endtime of block b_2 is $e_2 = 30$. However, b_2 cannot be deleted at e_2, but must remain in the chain until $t = 40$, which is the *endtime* of block b_3. Thus, the expiry time of b_2 is extended by 10 time units. If b_2 in the example has size 1MB and 10 extra seconds to remain in the chain, the RC of b_i is 10 MB·sec.

Limiting Block Height: The total RC of the chain will tend to increase as the total number of blocks. One way to alleviate this problem is to limit the number of edges coming from AOG. For that purpose, we propose K-height block insertion for some integer parameter K as follows. Suppose we wish to create a new block b_i, we create two edges: 1) one connecting to *ParentBlock* b_i^* and 2) the other to *PreviousBlock* b_i^-. Assume $b_i^* \neq b_i^-$. We propose the following: create an arrival ordering edge to b_i^- only if the height of b_i^* is less than K.

1) Suppose the height of b_i^* is less than or equal to K. Then the arrival ordering edge to b_i^- will be created as usual.
2) Suppose the height of b_i^* is greater than K. Then we do not create the edge to b_i^-, and RC will not incur for block b_i. Both `ParentBlockHash` and `PreviousBlockHash` are set to the hash of b_i^*.

The idea is that if the height of a block is more than K, we consider its height to be sufficient, and focus on minimizing the overall RC of the chain by removing the arrival ordering edge. Parameter K can range from 0 to ∞. $K = 0$ means that our blockchain is constructed based only on EOG, in which RC is always zero. $K = \infty$ means that the height of any block will be equal to the number of previously created blocks existing in the current chain, providing the maximum possible height to all the blocks.

Fig. 5 shows an example of blockchain with K. As K changes from 1 to 4, the height of b_6 increases to 4, 5, and 6 and the total RC increases to 20, 39, and 56. Clearly, there exists a tradeoff, reducing storage cost incurred by expiry extension versus providing greater height to the blocks for enhanced security. Thus, K is a tunable parameter, and we will explore how K affects the storage cost and average height in the simulation section.

E. Indefinite Lifetimes

It is possible that the lifetime of a transaction is not decided at the time of creation. Such transactions are referred to have *indefinite* lifetimes. We categorize transactions into three types according to their lifetime properties.

1) `Fixed:` The lifetime is known at the time of creation.

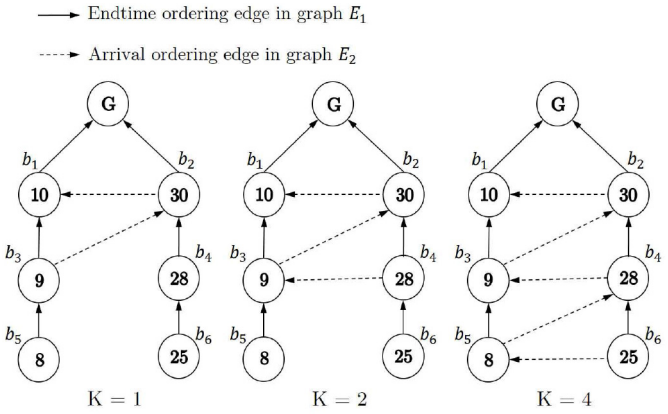

Fig. 5. Example with varying K.

2) `Permanent:` The lifetime is infinite at the time of creation.
3) `Indefinite:` The lifetime is not known, but can be expired at any time.

The deletion of indefinite-type transactions can be requested at any time. However, the transaction or its containing block is not deleted immediately upon request; such deletion would be inconsistent with the construction of LiTiChain. Instead, we assume that blocks containing indefinite-type transactions are *renewed* periodically as follows.

The renewal of a block is creating a new copy of the block and then deleting the block. The new block contains indefinite-type transactions copied from the original block. At the time of creation of a block, the lifetime of indefinite-type transactions are set to D, where D is a system parameter. Depending on applications, D may vary among indefinite-type blocks. Setting D too short would incur the overhead of frequent renewals, whereas setting D too long would result in the late deletion of the block. For example, D can be simply set on the order of days, months, or years, depending on the nature of associated transactions and can be determined by the entity which issues the deletion request. Suppose that the lifetime of a block expires and needs to be deleted. The block is checked if it contains indefinite-type transactions. If so, a new block with lifetime D is created containing the copy of those indefinite-type transactions, and the original block is deleted. It is possible that a request for deletion of a transaction may occur during the lifespan of the block. However, the block containing the transaction is not deleted until the next renewal of the block. Instead, at the next cycle of renewal, such transactions are excluded from the new block. This is consistent with the construction of LiTiChain. Besides, if there are multiple indefinite-type transactions with similar renewal times, those transactions can be grouped and put into a single indefinite-type block at the time of renewal.

In summary, the rules for renewal are as follows. Suppose a block is scheduled to be deleted at time t, then at time t.

1) Check if the block contains indefinite transactions which are not requested to be deleted.
2) If so, create a new block containing only those transactions and insert the block to blockchain. The block is scheduled to be renewed at $t + D$.
3) Delete the old block.

Algorithm 1 K-Height Insertion of a Block

Output : Newly created block b_i with endtime e_i
Given : renewal period D, height constraint K, transactions $\tau_1, \ldots, \tau_m$
/* Determining endtime e_i */
Set e_i be the maximum of the endtimes of $\tau_1, \ldots, \tau_m$. If some τ_j is of indefinite-type, regard its endtime as the current time plus D.
/* Determining the parent block according to endtime ordering */
$j^* \leftarrow \min\{j | e_i \leq e_j\}$
$b_i^* \leftarrow b_{j^*}$
Let d denote the height of parent block b_i^*.
/* Determining the previous block according to arrival ordering */
$l^* \leftarrow \max_{l=0,1,\ldots,i-1}\{l | \text{block } b_l \text{ exists in the chain}\}$
/* Constraining height of the new block */
if $d > K$ **then**
  $b_i^- \leftarrow b_i^*$
else
  $b_i^- \leftarrow b_{l^*}$
end if
Create block b_i. Add transactions $\tau_1, \ldots, \tau_m$ to b_i. In the header of b_i, do the following:
  Set timestamp field t_i to the current time.
  Set ParentBlockHash field to $Hash(b_i^*)$.
  Set PreviousBlockHash field to $Hash(b_i^-)$.
  Set ExpiryTime field to e_i.
Set DeletionTime variable for block b_i to e_i.

Block renewal can be used for maintaining the overall block height as follows. In case the lifetimes of blocks created earlier are short overall, and as a result, most of the intermediate nodes in AOG are deleted, the maximum distance paths from blocks to G will tend to be short, decreasing the overall block heights. This can be fixed by intermittently creating *bogus* blocks in order to increase the heights as follows. The bogus blocks contain empty transactions and have indefinite lifetimes. The system can track, e.g., the average maximum heights of blocks, and bogus blocks can be generated triggered by the event that the measured height falls below certain threshold. The purpose of bogus blocks is for keeping a sufficient number of blocks in the chain to maintain its overall height. The bogus blocks can be either renewed or deleted depending on the average maximum heights of the chain at the time of renewal.

IV. Algorithm

A. Block Creation and Insertion

The algorithm of K-height insertion of the ith block to the blockchain is outlined in Algorithm 1. The block format of the proposed blockchain is similar to that of conventional blockchain, except that the block header has following additional fields.

1) ParentBlockHash: Explained in Section III-C.

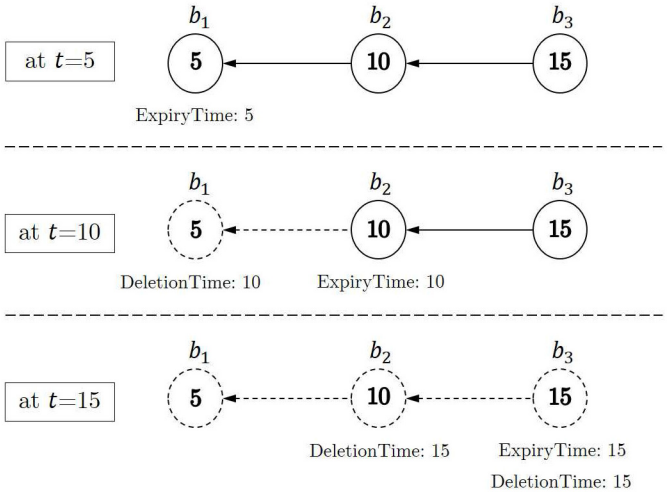

Fig. 6. Example of ExpiryTime and DeletionTime.

2) PreviousBlockHash: Explained in Section III-C.
3) ExpiryTime: The time of the expiration of the block lifetime.

ExpiryTime is set to the endtime of a block which is obtained by adding the lifetime of the block to the timestamp at which the block is created. A block will be deleted from the chain if ExpiryTime has passed, except that its expiry is extended by the next block. Thus, a node must separately maintain another variable for each block, which we call DeletionTime. DeletionTime of a block, say b_i, is initially set to the endtime e_i. If later block j is created, and b_j refers to b_i such that $b_i = b_j^-$ and $e_i < e_j$, DeletionTime field of b_i is extended to e_j, so that b_i is not deleted until the expiry of b_j. In summary, ExpiryTime represents the expiration of the transaction data within the block, whereas DeletionTime simply indicates when this block should be deleted. Note that only ExpiryTime is recorded at the block, but DeletionTime is a changing variable and thus is not recorded in the block because the block is not modifiable by definition.

An example is shown in Fig. 6. At $t = 5$, b_1 is expired without being deleted, because b_2 has reference to b_1 by $b_1 = b_2^-$ and $e_1 < e_2$. At $t = 10$, b_1 is deleted and b_2 is expired but is not deleted due to the reference by b_3. In case of b_3, because there is no block referring to b_3, expiry and deletion of b_3 occur simultaneously at $t = 15$.

B. Block Deletion and Renewal

Algorithm 2 shows the procedure to delete or renew a block. For every block, if its DeletionTime is reached, Algorithm 2 is executed. Note that if the current block to be deleted contains indefinite-type transactions which are not requested for deletion, a new block will be created containing a copy of those transactions. The timestamp of the new block is equal to that of the original block, since the timestamp represents the creation time of the transactions. The expiration and deletion time is set to D time units after; at the expiration, there will be the renewal of this new block, and the process repeats. Eventually, if all the transactions are requested for

Algorithm 2 Deletion/Renewal of a Block

1: Given : Block b_i to be deleted/renewed
2: **if** b_i contains indefinite-type transactions $\tau_1, \ldots, \tau_m$ which are not requested for deletion **then**
3: Create block b_j. Do K-height insertion of b_j with:
4: Add transactions $\tau_1, \ldots, \tau_m$ to b_j.
5: Set timestamp of b_j as t_i (timestamp of b_i).
6: Set `ExpiryTime` field and `DeletionTime` variable to current time plus D
7: **end if**
8: Delete b_i

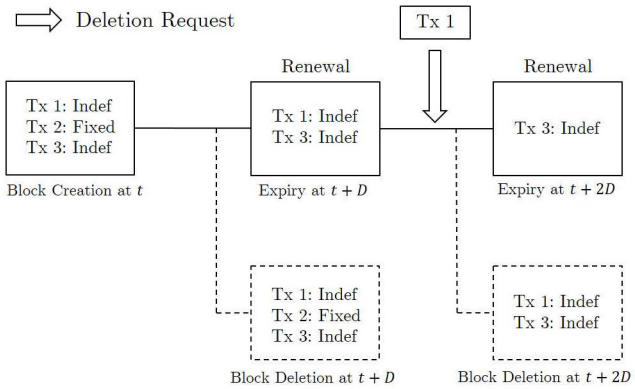

Fig. 7. Example of block renewal.

deletion, the block and transactions will be completely erased from the chain.

An example of block renewal is shown in Fig. 7. Suppose a block contains three transactions Tx 1, 2, and 3, where Tx 1 and 3 are of indefinite type and Tx 2 is of fixed type. The lifetime of Tx 1 and 3 is given by D, and suppose D is longer than the lifetime of Tx 2. The lifetime of the block is set to D and is scheduled to be deleted at time $t+D$. At time $t+D$, the lifetime of Tx 2 is expired and none of the indefinite-type transactions have been requested for deletion. Thus, the block is renewed and the new block contains the copy of Tx 1 and 3, and is scheduled to be deleted (renewed) at time $t+2D$. Then, before time $t+2D$, there was a request for deletion of Tx 1. Then at time $t+2D$, the block is renewed and only Tx 3 is included in the new block.

V. Analysis

In this section, we analyze how RC contributes to the total storage cost. Due to the dynamic nature of creation and deletion of blocks, it is difficult to analyze RC for general cases. Thus, for analytical tractability, we consider two cases: 1) the average cost analysis when the block creation times constitute a stationary Poisson process and 2) the worst-case analysis for a finite number of blocks. Throughout the analysis, it is assumed that each block has the unit size. We also assume that all the transactions have fixed-type lifetimes.

A. Average RC for Poisson Arrivals of Blocks

Suppose the blocks are created at time instants according to homogeneous Poisson process Π with rate λ. The lifetime of blocks is independent and identically distributed with probability density function (pdf) $g(\cdot)$. Let $\mathbb{E}^0$ denote the expectation with respect to the Palm distribution $\mathbb{P}^0$ with respect to the block arrival process Π. The Palm distribution [47] $\mathbb{P}^0$ is conditional on the "typical" block being created at time 0. The framework of Palm distribution is useful for evaluating the costs viewed from a typical block, i.e., informally speaking, a block randomly selected from the arrival process without bias.

Theorem 1: Let C denote the RC seen by a typical block from Π. We have that

$$\mathbb{E}^0[C] = \lambda \int_0^\infty \int_0^x \int_{x-y}^\infty (z-x+y)e^{-\lambda y} g(z)g(x)dz\ dy\ dx. \tag{1}$$

Proof: Suppose the lifetime of the typical block at $t=0$ is in $[x, x+dx)$ which occurs with probability $\approx g(x)dx$. Because RC depends only on the block arrived right next to the typical block but not on others. Thus, we consider the probability that the next block at $[y, y+dy)$ which is given by

$$\exp(-\lambda y) \cdot \lambda\ dy.$$

This is because there must be no arrival from Π during time interval $[0, y)$ which occurs with $\exp(-\lambda y)$, and the probability of arrival in $[y, y+dy)$ is given by λdy. Suppose that the lifetime of the next block is in $[z, z+dz)$, which occurs with probability $\approx g(z)dz$. The RC will incur if:

1) the arrival of the next block is earlier than the expiration of the lifetime of the typical block;
2) the lifetime of the next block is longer than the residual lifetime of the typical block.

The conditions are given by $y \le x$ and $z \ge x-y$, respectively. Because the block has unit size, the incurred RC is given by

$$\mathbf{1}(y \le x) \cdot \mathbf{1}(z \ge x-y) \cdot \{z-(x-y)\}$$

with the associated probability

$$g(x)dx \cdot \lambda \exp(-\lambda y)dy \cdot g(z)dz$$

which results in (1). ■

The RC can be regarded as the additional amount of time the block needs to stay at the chain. Assume that the mean lifetime is given by μ^{-1} for some $\mu > 0$. Thus, the mean occupation time of a typical block is given by

$$\mathbb{E}^0[C] + \frac{1}{\mu}.$$

The mean arrival rate of the blocks to the system is λ. Thus, by Little's formula, the mean storage space occupied by the chain is given by

$$\lambda\left(\mathbb{E}^0[C] + \frac{1}{\mu}\right)$$

which is the mean number of blocks in the chain in the steady state.

Theorem 1 enables us to evaluate the mean RC in closed form for some types of lifetime distributions. For example, suppose the lifetimes of blocks are independently and

exponentially distributed with parameter μ, that is

$$g(x) = \mu \exp(-\mu x), \ x \geq 0.$$

Then, we get from (1)

$$\mathbb{E}^0[C] = \begin{cases} \dfrac{1}{4\mu}, & \lambda = \mu \\ \dfrac{\lambda}{\mu - \lambda}\left[\dfrac{1}{\lambda + \mu} - \dfrac{1}{2\mu}\right], & \lambda \neq \mu. \end{cases}$$

B. Worst-Case RC

Next, we consider deterministic setup which yields the worst possible RC. The goal is to obtain an upper bound on the incurred RC. We consider the RC associated with a finite number of blocks during a finite-time horizon.

Assumption 1: Our assumptions for analysis are as follows.

1) A total of n blocks $b_1, \ldots, b_n$ with endtimes $e_1, \ldots, e_n$ are created in sequence.
2) The blocks have distinct endtimes.
3) The last block b_n is created before the earliest endtime among all the blocks.

Due to Assumption 1 3), no blocks are deleted until all of the n blocks are created. Also from the assumptions, RC is completely determined by the ordering of $e_1, \ldots, e_n$. We consider the ordering for the worst possible RC, which gives an upper bound of RC for all possible cases. Below, we formulate the worst-case analysis as the combinatorial optimization problem.

We will consider the total storage costs for cases of $K = \infty$ and $K = 0$. We will compare two cases, because evaluating case $K = \infty$ will give a full account of RC in our scheme, whereas no RC will be incurred under case $K = 0$. Let $N_\infty(t)$ and $N_0(t)$ denote the total number of blocks in the blockchain at time t with $K = \infty$ and $K = 0$, respectively. Assume b_1 is created at time 0, and let T denote the latest endtime among $e_1, \ldots, e_n$. We would like to compare the total storage cost incurred during the lifetime of the chain. Because the blocks have unit size, the total storage cost for case $K = \infty$ is given by

$$\int_0^T N_\infty(t)dt$$

that is, it is the area under $N_\infty(t)$ by definition. Similarly, we consider $\int_0^T N_0(t)dt$ which is also equal to the sum of the lifetime of the blocks. The total RC of case $K = \infty$ is then given by

$$\int_0^T [N_\infty(t) - N_0(t)]dt$$

for which we will find an upper bound.

We introduce the following notations. Denote the earliest endtime by $\hat{e}$. Consider set S such that

$$S \coloneqq \{e_1 - \hat{e}, e_2 - \hat{e}, \ldots, e_n - \hat{e}\}.$$

Consider $\lceil (n/2) \rceil$ smallest numbers of S and denote the ith smallest number by s_i. Next, consider the rest of numbers from S and denote the ith smallest number by l_i. We have the following relation among s_i's and l_i's:

$$0 = s_1 < s_2 < \cdots < s_{\lceil \frac{n}{2} \rceil} < l_1 < l_2 < \cdots < l_{\lfloor \frac{n}{2} \rfloor}.$$

Note that strict inequalities hold because endtimes are distinct.

Theorem 2: We have that

$$\int_0^T [N_\infty(t)dt - N_0(t)]dt \leq \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (l_i - s_i).$$

We prove the theorem by following the proof outlined in [48] which proposes the string reversal algorithm which will be explained in the sequel. Define $f(x) \coloneqq \max(x, 0)$.

Lemma 1: For any real numbers $s_1, s_2 < l_1, l_2$

$$f(l_1 - s_1) + f(l_2 - s_2) > f(l_1 - l_2) + f(s_1 - s_2).$$

Proof: The proof for Lemma 1 is similar to proof of [48, Lemma 1]. For $s_1, s_2 < l_1, l_2$, we have inequalities

$$\begin{aligned} l_2 &> s_1 \\ (l_2 - s_2) &> (s_1 - s_2) \\ (l_1 - s_1) &> (l_1 - l_2) \\ (l_1 - s_1) + (l_2 - s_2) &> (l_1 - l_2) + (s_1 - s_2). \end{aligned} \tag{2}$$

From (2), we can show the following in four cases.

1) If $l_1 > l_2$ and $s_1 > s_2$

$$f(l_1 - s_1) + f(l_2 - s_2) > f(l_1 - l_2) + f(s_1 - s_2).$$

2) If $l_1 > l_2$ and $s_1 \leq s_2$

$$f(l_1 - s_1) + f(l_2 - s_2) > f(l_1 - l_2).$$

3) If $l_1 \leq l_2$ and $s_1 > s_2$

$$f(l_1 - s_1) + f(l_2 - s_2) > f(s_1 - s_2).$$

4) If $l_1 \leq l_2$ and $s_1 \leq s_2$

$$f(l_1 - s_1) + f(l_2 - s_2) > 0.$$

From (2) and the above four cases, Lemma 1 is proved. ■

Let X be an ordered sequence of n numbers $x_1, \ldots, x_n$. Define

$$C(X) = \sum_{i=1}^{n-1} f(x_{i+1} - x_i).$$

Lemma 2: Consider sequence X that is an arbitrary permutation of $s_1, s_2, \ldots, l_1, l_2, \ldots,$. We can find permutation X^* which maximizes $C(X)$ by successively applying a string reversal algorithm [48] to X. We have that

$$C(X^*) = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (l_i - s_i).$$

Proof: We will follow similar arguments as those of [48, Lemma 2]. We consider the even and odd cases of n.

Case 1 ($n = 2k$): Consider X_1 which is an arbitrary permutation of $s_1, s_2, \ldots, l_1, l_2, \ldots$ Suppose we apply the string reversal algorithm to X_1, change it to X_2 as follows. We say X is in *alternating arrangement* [48] if the odd-indexed elements

of X or $x_1, x_3, x_5, \ldots$ is some permutation of small values $s_1, s_2, \ldots$ and the even-indexed elements of X are a permutation of $l_1, l_2, \ldots$ Suppose X_1 is *not* in alternating arrangement, we regard X_1 as a string and operate on a substring of X_1 which is not alternating such that

$$X_1 = , \ldots, s_3, [s_4, \ldots, l_2], l_3, \ldots$$
$$X_2 = , \ldots, s_3, [l_2, \ldots, s_4], l_3, \ldots$$

That is, the substring in the bracket of X_1 is reversed in order to yield the substring in the brackets in X_2. We can write $C(X_1)$ and $C(X_2)$ as follows:

$$C(X_1) = \Delta + f(s_4 - s_3) + f(l_3 - l_2)$$
$$C(X_2) = \Delta + f(l_2 - s_3) + f(l_3 - s_4)$$

where Δ means the common cost for X_1 and X_2. By Lemma 1, we have that $C(X_2) > C(X_1)$.

This implies that we can successively apply string reversal algorithm to X and increase the cost function $C(X)$ until X is in alternating arrangement. Due to the property of f, it is clear that all strings X in alternating arrangement have the same cost. Specifically, consider $X^* = s_1, l_1, s_2, l_2 s_3, l_3, \ldots, s_{k-1}, l_{k-1}, s_k, l_k$. We have that for arbitrary string X

$$\begin{aligned} C(X) &\le C(X^*) \\ &= (l_1 - s_1) + (s_2 - l_1) + (l_2 - s_2) \\ &\quad + \cdots + (s_k - l_{k-1}) + (l_k - s_k) \\ &= \sum_{i=1}^{k} l_i - \sum_{i=1}^{k} s_i. \end{aligned}$$

Case 2 ($n = 2k+1$): The proof is nearly identical to case 1. We can show that the maximum cost is achieved by alternating arrangement of string. The maximum cost can be achieved by string $X^* = s_1, m, l_1, s_2, l_2 s_3, l_3, \ldots, s_{k-1}, l_{k-1}, s_k, l_k$, where m is the $k+1$th largest number, or the element in the middle of string $s_1, s_2, \ldots, l_1, l_2, \ldots$ We have that

$$\begin{aligned} C(X) &\le C(X^*) \\ &= (m - s_1) + (l_1 - m) + (l_2 - s_2) \\ &\quad + \cdots + (l_{k-1} - s_{k-1}) + (l_k - s_k) \\ &= \sum_{i=1}^{k} l_i - \sum_{i=1}^{k} s_i. \end{aligned}$$

■

Proof of Theorem 2: The RC for case $K = \infty$ is bounded above as follows. By definition, the RC of block i is given by $\max(e_{i+1} - e_i, 0) = f(e_{i+1} - e_i)$. Thus, the total RC is given by

$$C = \sum_{i=1}^{n-1} f(e_{i+1} - e_i).$$

Note $\int_0^T N_0(t)dt$ is simply the sum of the lifetimes of the blocks. Also due to Assumption 1, we have that $\int_0^T N_\infty(t)dt = C + \int_0^T N_0(t)dt$. From Lemma 2, we have that

$$C \le \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (l_i - s_i).$$

The upper bound is achieved if $e_1, \ldots, e_n$ are in alternating arrangement such that endtimes with odd indices $e_1, e_3, e_5, \ldots$ are given by some permutation of small endtimes $\hat{e} + s_1, \hat{e} + s_2, \hat{e} + s_3, \ldots$, and the endtimes with even indices are given by some permutation of large endtimes $\hat{e} + l_1, \hat{e} + l_2, \hat{e} + l_3, \ldots$ ■

We provide a looser bound on the ratio of storage costs when n is even. We show that the storage overhead due to RC is at most twice the storage cost without RC.

Corollary 1: Suppose n is even. We have that

$$\frac{\int_0^T N_\infty(t)dt}{\int_0^T N_0(t)dt} \le \frac{2\sum_{i=1}^{k} l_i}{\sum_{i=1}^{k}(s_i + l_i)} \le 2.$$

Proof: Let $n = 2k$. Clearly we have that

$$\int_0^T N_0(t)dt = n\hat{e} + \sum_{i=1}^{k}(s_i + l_i)$$

which implies that, by Theorem 2

$$\begin{aligned} \int_0^T N_\infty(t)dt &\le n\hat{e} + \sum_{i=1}^{k}(s_i + l_i) + \sum_{i=1}^{k}(l_i - s_i) \\ &= n\hat{e} + 2\sum_{i=1}^{k} l_i. \end{aligned}$$

Thus, we have that

$$\begin{aligned} \frac{\int_0^T N_\infty(t)dt}{\int_0^T N_0(t)dt} &\le \frac{n\hat{e} + 2\sum_{i=1}^{k} l_i}{n\hat{e} + \sum_{i=1}^{k}(s_i + l_i)} \le \frac{2\sum_{i=1}^{k} l_i}{\sum_{i=1}^{k}(s_i + l_i)} \\ &\le \frac{2\sum_{i=1}^{k} l_i}{\sum_{i=1}^{k} l_i} = 2 \end{aligned}$$

for which we used $\hat{e} \ge 0$ and $l_i > s_i \ge 0$ for all i. ■

VI. Performance Evaluation

In this section, we evaluate the performance of our blockchain system through simulation. We focus on evaluating average storage occupied by blocks and average block heights.

A. Performance Metrics

As previously, let $N(t)$ denote the number of blocks in the chain at time t. We assume all the blocks have the unit size. Thus, the average storage space occupied by the chain is proportional to $N(t)$. We define the time average of storage occupied by the chain by

$$\overline{N} := \lim_{T \to \infty} \frac{1}{T} \int_0^T N(t)dt.$$

We define the longest path from the newly created block to the genesis block as the height of a block. The average height $\overline{D}$ is defined as taking the average block heights over all blocks in the chain measured at the instants of block creation and deletion.

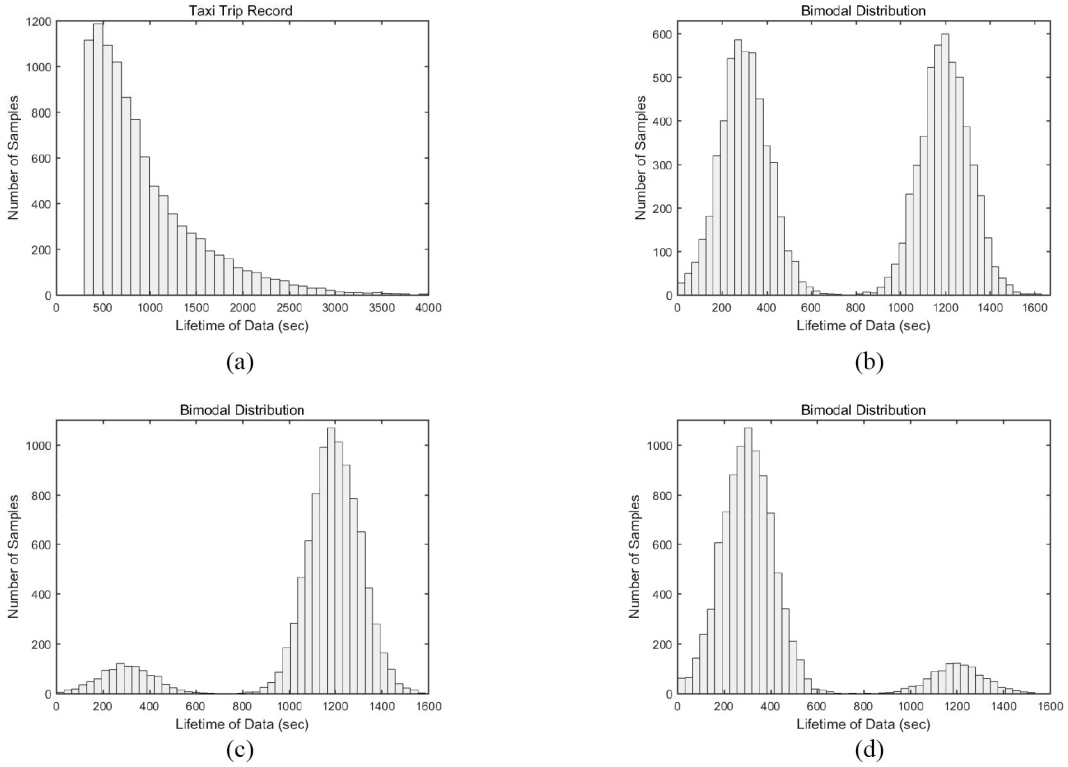

Fig. 8. Distribution of lifetime. (a) Operation time distribution from taxi trip record. (b) Distribution of Z. (c) Distribution of $\hat{Z}$. (d) Distribution of $\tilde{Z}$.

B. Simulation Settings

We used the MATLAB tool to implement the proposed algorithms and to perform simulation for three different types of lifetime distributions. In the simulation, only fixed-type transactions were considered.

Case 1: In order to use realistic IoT data with lifetimes, we chose the data set published by the New York City Taxi and Limousine Commission (TLC) on trip record data for yellow taxis in December 2018 [49]. We extracted 10 000 data points of operating times ranging between 301 and 4000 s from the population of 8 million data points. The operating time is the time difference between the engagement and disengagement of the taximeter for a passenger group.

In addition to the taxi data set, we assumed that each taxi will generate some event data every 1 s during a trip. This event data can be considered as monitoring information, such as real-time taxi fare, location, and speed. Indeed, if autonomous taxi vehicles become prevalent in the near future, such real-time monitoring data of vehicles will be generated in high volume. We assume such event data is reported to the edge servers. The event data is considered as "transactions" which will be stored in our blockchain in the simulation. We assume that a block is created every 5 min; thus, a block contains 300 transactions. We also assume that the size of each block is 1 MB. In our blockchain, the trip record data and event data are used as the transactions to be stored. The operating time is defined as the lifetime of those transactions. Fig. 8(a) shows a distribution of operation times from the taxi trip record.

Case 2: We hypothesize that if the lifetime has a bimodal distributions with modes at small and large lifetime values, i.e., lifetimes are either very short or very long, it would worsen the cost overhead due to RC. The rationale is that the blocks with short lifetime suffer relatively more from the amount of time held back by blocks with long lifetimes. In order to obtain lifetime data with a bimodal distribution, we produced lifetime data sampled from Z which is a mixture of two Gaussian distributions as follows:

$$Z = \begin{cases} Z_1, & \text{w.p. } 0.5 \\ Z_2, & \text{w.p. } 0.5 \end{cases}$$

where $Z_1 \sim N(300, 110^2)$ and $Z_2 \sim N(1200, 110^2)$. We sampled 10 000 lifetime data from Z. Fig. 8(b) shows the distribution of Z.

Case 3: In order to compare the RC generated by the change in the mixing proportion in the Gaussian mixture, case 3 has the same settings as case 2 except for the mixing proportions. We consider two cases of lifetime distributions as follows:

$$\hat{Z} = \begin{cases} Z_1, & \text{w.p. } 0.1 \\ Z_2, & \text{w.p. } 0.9 \end{cases} \tag{3}$$

$$\tilde{Z} = \begin{cases} Z_1, & \text{w.p. } 0.9 \\ Z_2, & \text{w.p. } 0.1. \end{cases} \tag{4}$$

Fig. 8(c) and (d) shows the distribution of (3) and (4), respectively.

C. Simulation Results

Fig. 9(a)–(c) shows the simulation results for trip record data of New York taxi (case 1). Fig. 9(d)–(f) shows the results for

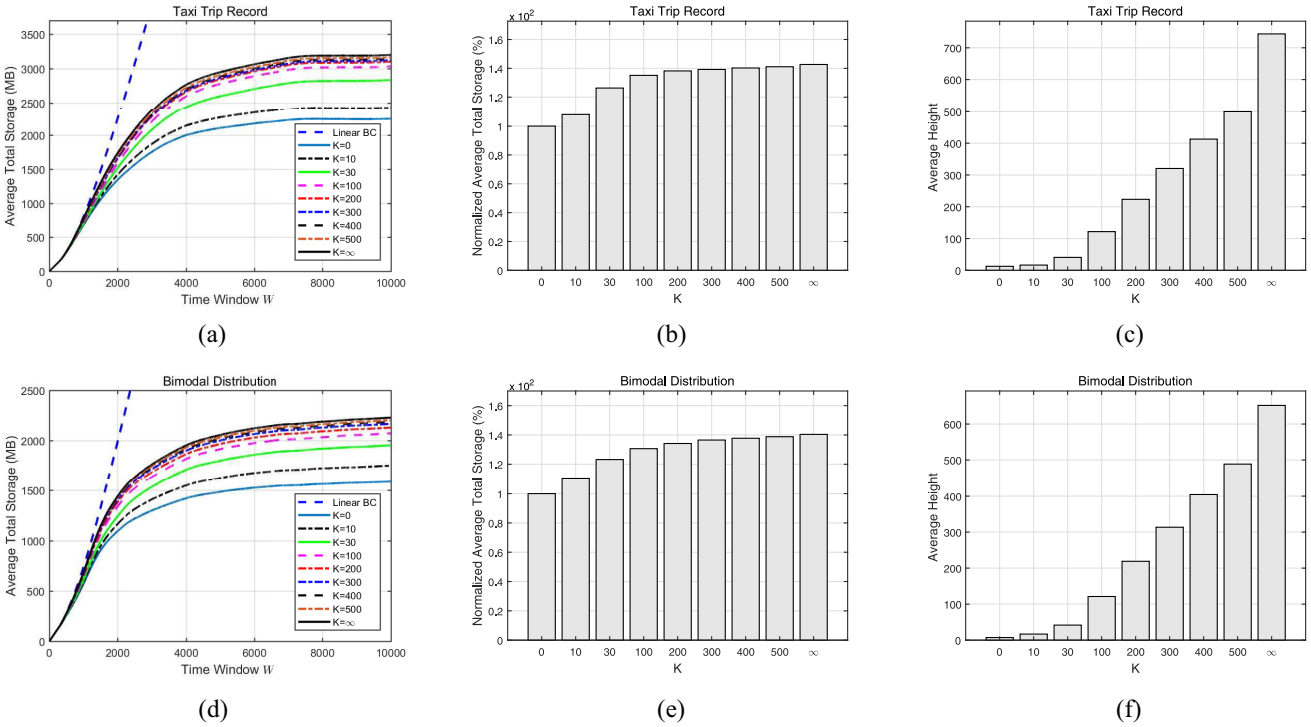

Fig. 9. Average total storage, total storage cost ratio, and height under LiTiChain.

bimodally distributed lifetime (case 2). Fig. 9(a) and (d) shows the average storage cost against varying values of height constraint K. Fig. 9(a) and (d) shows the plots of storage occupied by blocks averaged over time interval $[0, W]$ against window value W. We observe that the average storage blows up for conventional blockchain because it permanently stores all the incoming transactions. However, for our schemes, the average storage converges quickly to stable values.

Fig. 9(b) and (e) shows the average storage $\overline{N}$ for varying height constraint K. The baseline case is $K = 0$, and the average storage for changing K is normalized to the baseline in the figures. Fig. 9(b) and (e) shows the average storage are about 100%–142% and 100%–140% of the baseline storage depending on K. Note that Fig. 9(b), (c), (e), and (f) is plotted against a set of K values, and their horizontal axis is not to scale. By controlling height constraint K from 0 to ∞, we achieve varying degrees of the tradeoff between security and storage costs. As K becomes larger, the average height of the blocks increases, strengthening the security of the chain. With smaller values of K, the block heights tend to be shallow, however, the overall RC will decrease, leading to a reduced storage requirement for the blockchain.

Let us compare the performances of cases 1 and 2. We observe that the overhead due to RC tends to be similar for cases 1 and 2. For case 2, we may consider the following back-of-the-envelop calculation on the overhead due to RC. Suppose the lifetime can take only two values: 1) very long and 2) very short. Assume that the probability of a block having either length of lifetime is 0.5. Then roughly half of the total blocks will be created with the short lifetime. Furthermore, roughly half of those will be referred to by a block with the long lifetime which arrived next to it. Thus, the blocks with originally short lifetime, but are changed to have the long lifetime, will take about 25% of the total blocks. This is 50% increase in the proportion of the blocks with a long lifetime. Assuming the blocks with the long lifetime will occupy most of the total storage, the overhead of RC will be close to 50% of the baseline storage. From Fig. 9(e), we indeed observe that the overhead due to RC rises over 40% if $K = \infty$. Interestingly, similar overhead is observed in Fig. 9(b) with lifetime distribution with single tail given by Fig. 8(a). This seems to be because most of the blocks have a short lifetime from the shape in Fig. 8(a). Thus, if blocks with short lifetimes are referred to by a few blocks with long lifetime, the increased overhead in the RC is expected to be significant to the overall storage.

Blocks Generated in Bursts: Next, we consider the case where blocks are generated in bursts. Specifically, we evaluate the storage cost of a finite number of, say n, blocks, where all the n blocks are created during a short time period relative to their lifetimes. Note that this setup is similar to Assumption 1 in the RC analysis. Our goal is to compare the RC from worst case analysis with the RC from the previous data sets. We proved that the maximum RC incurs if the endtimes are in alternating arrangement. In the simulation, we compare the theoretical worst-case cost with the cases the endtimes of blocks are in a random arrangement according to the data sets.

In the simulation, 50 blocks are created in burst, where the block lifetimes are sampled from the distributions of cases 1 and 2. We assume that the blocks are created at distinct time instants but within a negligibly short period of time, so that the order of block endtimes is equal to that of the lifetimes. For each case, the storage costs for $K = 0$ and $K = \infty$

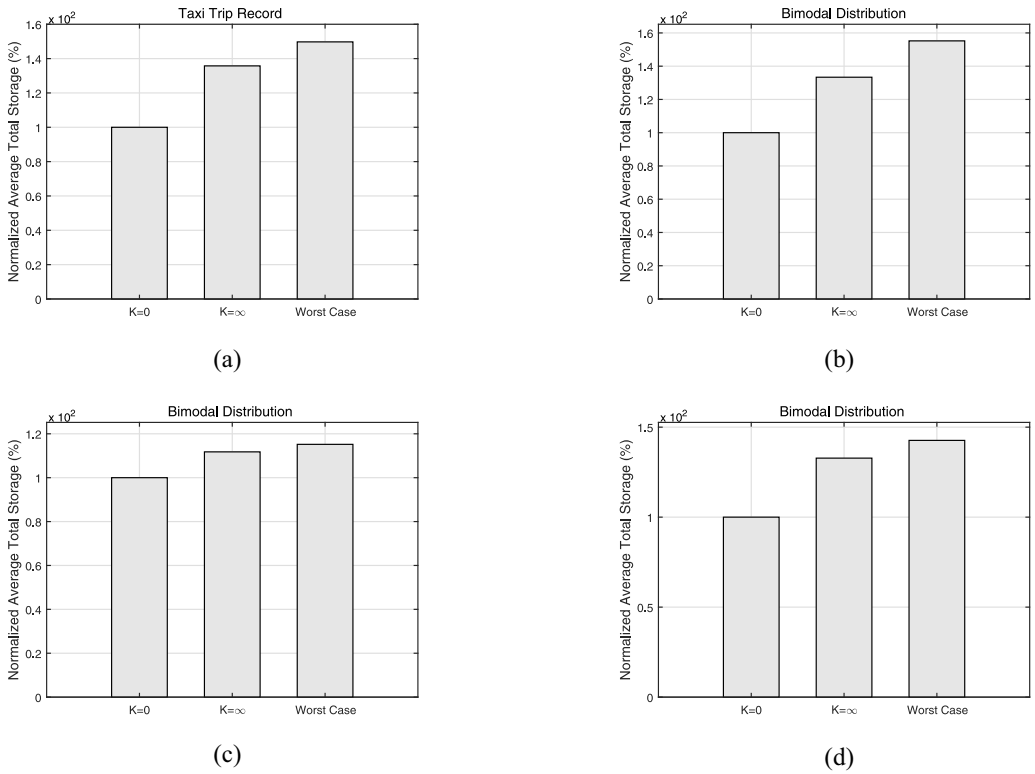

Fig. 10. Average total storage cost ratio under burst data.

are compared with the worst case in which the lifetimes are reordered such that they are exactly in alternating arrangement.

Fig. 10(a) and (b) shows the comparison for the data sets in cases 1 and 2, respectively. The storage costs are averaged over 200 times for each plot. The cost ratio of $K = \infty$ (resp. the worst case) relative to baseline with no RC ($K = 0$) is about 136% (resp. 149%) in case 1. For case 2, the ratio is similar, given by 133% and 155%, respectively. The gap of cost ratio between $K = \infty$ and the worst case is higher for bimodal distributions (13% versus 21%). This shows that indeed with bimodal distribution in which short-lived blocks are held back by long-lived blocks, the worst case with alternating arrangement can do much worse than randomized arrangement. In reality, it is unlikely that the alternating arrangement occurs by chance; nonetheless, it is useful to know the upper bound on the additional costs required to store the blockchain in its entirety.

Next, we examine how the RC is affected by the change in the mixing proportion of the Gaussian mixture in the lifetime distribution. Fig. 10(c) and (d) shows the storage cost ratio for (3) and (4) in case 3, respectively. The cost ratio of case $K = \infty$ (resp. the worst case) to case $K = 0$ is about 112% (resp. 115%) for (3). For (4), the cost ratio is given by 133% and 143%, respectively. We see that the relative overhead due to RC is in the order of (b) > (d) > (c) for the distributions in Fig. 10. The result can be explained as follows. In case of distribution, Fig. 10(c), because most blocks have long lifetimes, and thus most short-lived blocks are likely to be retained by long-lived blocks. However, the proportion of short-lived blocks is small in the first place; thus the effect of RC on the total storage cost is relatively low. In case, Fig. 10(d), there are many blocks with short lifetime, and there will be a few blocks held back by long-lived blocks. However, if they get retained by long-lived blocks, the relative increase in the storage cost is high. The ratio of retained short-lived blocks and its effect on the total storage cost is most "balanced" in the case of Fig. 10(c), i.e., at the mixing proportion of 0.5 as in case 2, which yields the highest overhead due to the RC.

VII. Conclusion

In this article, we proposed LiTiChain, a blockchain with finite-lifetime blocks, which is suitable for edge-based IoT systems. Through LiTiChain, we address the problem of conventional blockchain such that it is a permanently growing list of information chunks. The issue of storing long blockchains at full nodes can be resolved, if the outdated information can be removed in a timely manner. We introduced a tree-structured EOG which organizes the blocks according to their endtimes, so that the blocks can be deleted in an orderly fashion while maintaining the connectivity of the chain. In order to achieve various degrees of tradeoff between RC and the overall block heights, we proposed K-height block insertion. We also proposed the block renewal methods so as to deal with transactions with indefinite lifetimes. The simulation results demonstrated the effect of the aforementioned tradeoff on the total storage costs depending on the properties of lifetime distributions. Future work will include devising a permissionless version of erasable blockchains with insertion/deletion throughputs acceptable for the IoT systems.

REFERENCES

[1] (Jan. 2019). *IoT Report: How Internet of Things Technology Growth Is Reaching Mainstream Companies and Consumers*. [Online]. Available: https://www.businessinsider.com/internet-of-things-report

[2] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the Internet of Things: Research issues and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

[3] S. K. Lo *et al.*, "Analysis of blockchain solutions for IoT: A systematic literature review," *IEEE Access*, vol. 7, pp. 58822–58835, 2019.

[4] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[5] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A comprehensive survey of blockchain: From theory to IoT applications and beyond," *IEEE Internet Things J.*, to be published.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008.

[7] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[9] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[10] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[11] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[12] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[13] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 60–66, Oct. 2016.

[14] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized IoT data management using blockchain and trusted execution environment," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, 2018, pp. 15–22.

[15] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. ACM 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2014, pp. 68–81.

[16] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, 2015, pp. 73–78.

[17] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[18] J. Pan, J. Wang, A. Hester, I. AlQerm, Y. Liu, and Y. Zhao, "EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4719–4732, Jun. 2019.

[19] M. A. Rahman, M. M. Rashid, M. S. Hossain, E. Hassanain, M. F. Alhamid, and M. Guizani, "Blockchain and IoT-based cognitive edge framework for sharing economy services in a smart city," *IEEE Access*, vol. 7, pp. 18611–18621, 2019.

[20] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of IoT data," in *Proc. ACM Cloud Comput. Security Workshop*, 2017, pp. 45–50.

[21] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, Vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[22] S. Meiklejohn *et al.*, "A fistful of bitcoins: Characterizing payments among men with no names," in *Proc. ACM Conf. Internet Meas.*, 2013, pp. 127–140.

[23] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[24] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)*, Eur. Parliament Council Eur. Union, Brussels, Belgium, May 2016.

[25] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "Blockchain: A distributed solution to automotive security and privacy," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 119–125, Dec. 2017.

[26] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[27] J. Wan, J. Li, M. Imran, D. Li, and F. E. Amin, "A blockchain-based solution for enhancing security and privacy in smart factory," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3652–3660, Jun. 2019.

[28] E. Zaghloul, T. Li, and J. Ren, "An attribute-based distributed data sharing scheme," in *Proc. IEEE Global. Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[29] H. R. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018.

[30] J. Benet and N. Greco. (2017). *Filecoin: A Decentralized Storage Network*. [Online]. Available: https://filecoin.io/filecoin.pdf

[31] W. Tong, X. Dong, Y. Shen, and X. Jiang, "A hierarchical sharding protocol for multi-domain IoT blockchains," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[32] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," White Paper, pp. 1–47, 2017. [Online]. Available: https://plasma.io/plasma.pdf

[33] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.

[34] J. Kang *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.

[35] T. Jiang, X. Chen, and J. Ma, "Public integrity auditing for shared dynamic cloud data with group user revocation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363–2373, Aug. 2016.

[36] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2017, pp. 468–475.

[37] C. Cai, X. Yuan, and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–7.

[38] Y. Zhu, G. Zheng, and K.-K. Wong, "Blockchain empowered decentralized storage in air-to-ground industrial networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3593–3601, Jun. 2019.

[39] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in IoT using blockchain," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2017, pp. 261–266.

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

[41] T. Jiang, H. Fang, and H. Wang, "Blockchain-based Internet of Vehicles: Distributed network architecture and performance analysis," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4640–4649, Jun. 2019.

[42] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep./Oct. 2019.

[43] J. Xu *et al.*, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8770–8781, Oct. 2019.

[44] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," in *Proc. IEEE Symp. Security Privacy (SP)*, May 2019, pp. 124–138.

[45] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based," *IACR Cryptol. ePrint Archive*, vol. 2019, pp. 406–457, 2019.

[46] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.

[47] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. New York, NY, USA: Springer, 2007.

[48] G. Cohen, E. Tonkes, and G. Coast, "Dartboard arrangements," *J. Comb.*, vol. 8, no. 2, pp. 1–8, 2001.

[49] *TLC Trip Record Data for Yellow Taxi Trip Records*, New York City Taxi Limousine Commission, Silver Spring, MA, USA, Dec. 2018. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Chan Kyu Pyoung received the B.S. degree in information and telecommunication engineering from Incheon National University, Incheon, South Korea, in 2010, and the M.S. degree in computer and radio communications engineering from Korea University, Seoul, South Korea, in 2012, where he is currently pursuing the Ph.D. degree.

His research interests include network resource management in cloud and edge, network virtualization, software-defined networking, and blockchain system.

Seung Jun Baek (Member, IEEE) received the B.S. degree from Seoul National University, Seoul, South Korea, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, TX, USA, in 2002 and 2007, respectively.

From 2007 to 2009, he was a Member of Technical Staff with the Communications and Medical Systems Laboratory, DSP Systems R&D Center, Texas Instruments, Dallas, TX, USA. In 2009, he joined the College of Informatics, Computer Science and Engineering Department, Korea University, Seoul, where he is currently a Full Professor. His research interests include mobile computing systems, machine learning, and data-driven optimization.