IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 13, 2020, accepted December 10, 2020, date of publication December 15, 2020,
date of current version December 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3044991

SELCOM: Selective Compression Scheme for Lightweight Nodes in Blockchain System

TEASUNG KIM[1], SEJONG LEE[2], YONGSEOK KWON[2], JAEWON NOH[1], SOOHYEONG KIM[2], AND SUNGHYUN CHO[2], (Member, IEEE)

[1]Department of Computer Science and Engineering, Hanyang University, Ansan 15588, South Korea
[2]Department of Computer Science and Engineering, Major in Bio Artificial Intelligence, Hanyang University, Ansan 15588, South Korea

Corresponding author: Sunghyun Cho (chopro@hanyang.ac.kr)

This work was supported in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT), Research on ICT core technologies and cultivation of innovative talents to lead next-generation smart healthcare under Grant 2019-0-01601, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1D1A1B07049043.

ABSTRACT With the development of blockchain technology, participants need to have huge storage volumes to deal with the growing blockchain ledger size over time. This requirement leads to the conditional participation and verification of participants, thus weakening the decentralization of a blockchain system. Several compression schemes have been proposed to mitigate this storage problem by compressing a blockchain ledger based on redundancy, modular functions, and hash functions. However, these schemes have the limitation of accumulating the compression results to validate the retained blocks. The accumulation gradually reduces the storage volume for the blockchain ledger within the storage volume of nodes with limited resources, thus reducing the verification capability of the nodes. In this paper, a selective compression scheme using a checkpoint-chain is proposed to prevent the accumulation of compression results. The checkpoint-chain is a second blockchain that stores the checkpoints compressing existing blocks through a block Merkle tree. An update process is also proposed to prevent the accumulation of checkpoints by combining them. As numerous blocks can be verified with only a few updated checkpoints, blockchain nodes with limited resources can reduce the storage volume for the blockchain ledger and achieve high verification capabilities. Finally, compared with the existing compression schemes, the proposed scheme can achieve an average reduction in the storage overhead and an average increase in the verification capability of 76.02% and 13.90%, respectively. Moreover, the corresponding performance improvements are 86.14% and 15.44% when the update process is performed, respectively.

INDEX TERMS Blockchain, block Merkle tree, second blockchain, storage efficiency.

I. INTRODUCTION

Blockchain technology has emerged as a major approach for implementing distributed systems [1]. It has been applied to various systems, such as logistics [2], [3], distribution [4], [5], notarial [6], [7], and medical [8], [9] systems. A blockchain is a shared ledger that allows all the participants in a distributed system to maintain the same states of data based on cryptography and consensus algorithms. It consists of a set of serial blocks, and each block has a set of serial transactions. Blocks and transactions are the historical records of the creation and change of data in a blockchain system. They are robust against falsification because both cryptography and consensus algorithms are used to ensure the integrity and consistency of blockchain. Moreover, blockchain can guarantee various advantages, such as decentralization, anti-forgery, and traceability. Therefore, blockchain technology has become essential for implementing traditional centralized models into distributed systems.

The blockchain technology has a typical storage problem owing to the growing interest in its application to various systems [10]–[12]. The volume of a blockchain increases over time owing to the continuous addition of new blocks when participants generate new data in a system. For example, the cryptocurrency systems with initial blockchains, Bitcoin and Ethereum, have grown to require huge storage volumes of 260 GB and 120 GB by early 2020, respectively [13]. As the blockchain technology expands to other

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wu.

systems, the blockchain data have become not only simple bank account data but also real-world data, such as electronic health records [14], industrial data [15], and video data [16]. These real-world data, which are larger than transaction data, form a blockchain that requires a larger storage volume. This requirement of a huge storage volume is a major bottleneck for operating the blockchain technology on devices with limited resources, such as sensors and Internet-of-things devices [17]. Furthermore, this requirement undermines the free participation and verification of a blockchain system, which weakens the decentralization of the system.

Various schemes have been proposed to solve the storage problem that can be classified into summarization and compression schemes. Some summarization schemes have been proposed using a recursive summarization tree [18], account tree [19], and snapshot with unspent transactions outputs (UTXOs) [20]. These schemes can solve the storage problem by replacing blockchain data with a summary. However, existing summarization schemes cannot be used to real-world data because they only focused on bank account data in cryptocurrency systems, such as balances and empty addresses [18], [19]. In contrast, compression schemes can be used in various systems with real-world data, as they can be applied strings and structures of blockchain [21], [22]. Various compression schemes have been proposed based on string redundancy [21], [22], the residue number system (RNS) [23], and hash functions [20], [24]. The key aspect of compression schemes is that smaller-sized compression results can replace blockchain data, and the results are used to recover or validate the data. Moreover, they not only address the storage problem but also enable independent verification of each block or transaction.

However, existing compression schemes have the limitation of accumulating compression results or the necessary proofs for verification as a blockchain grows. For example, a scheme using a compressible blockchain [20] generates a snapshot and stores it in a second blockchain. However, snapshots accumulate as the number of blocks increases over time. A drawback of the compression schemes is that the compression results cannot be used as actual blockchain data and are used to perform the verification of the data. Verification of the original data requires performing a recovery (decompression) process or maintaining the copied original data separately. Thus, accumulating compression results reduces the verification capability of blockchain nodes with limited resources. If only some blockchain nodes have verification capabilities, the decentralization of the blockchain system can be weakened. Therefore, as all the blockchain nodes must be able to achieve their maximum verification capabilities, the long-term accumulation of compression results must be prevented.

We propose a selective compression (SELCOM) scheme using a block Merkle tree (BMT) to solve the storage problem and achieve a higher verification capability without accumulations. The BMT is an extended version of the Merkle tree proposed in our previous study that stores blocks instead of transactions [25]. In SELCOM, the blocks in a blockchain are compressed into a checkpoint. Subsequently, the checkpoints are used to construct a checkpoint-chain, which is a second blockchain. Then, the compressed blocks can be selectively erased or maintained based on the individual decisions of the blockchain nodes. Moreover, we propose an update process in SELCOM to avoid the accumulation of a second blockchain and resolve the drawback of our previous study where an unbalanced BMT was created. The update process combines accumulated checkpoints into fewer checkpoints. In conclusion, we analyze the proposed and existing schemes and show that our proposed scheme achieves a lower storage overhead and higher verification capabilities under the same storage limits as compared with the existing schemes.

The main contributions of this paper are as follows:

1) We propose a selective compression scheme, called SELCOM, to solve the storage problem for blockchain nodes with limited resources. The proposed scheme offers efficient storage volume management by allowing each node to maintain blocks selectively through the proposed checkpoint-chain, which is a second blockchain.
2) We design an update process in SELCOM to mitigate the limitation of reduced verification capability due to the accumulation of compression results in existing compression schemes. The update process combines checkpoints in a checkpoint-chain. Under the same storage limits, blockchain nodes can achieve higher verification capabilities because a small number of checkpoints can be used to validate numerous blocks.
3) We compare the numerical results of the proposed and existing compression schemes. In particular, we calculate the storage overhead required for independent verification guaranteed by each compression scheme. Compared with the existing schemes [20], [26], SELCOM can achieve an average reduction in the storage overhead and an average increase in the verification capability of 76.02% and 13.90%, respectively. Moreover, the corresponding performance improvements are 86.14% and 15.44% when an update process is performed, respectively.

The rest of this paper is organized as follows. Section II provides a review of existing compression schemes and Section III provides preliminaries. Section IV provides a detailed description of our proposed scheme. Section V presents the numerical results of the analysis of the proposed and existing schemes. Finally, Section VI presents the conclusions.

II. RELATED WORKS

In this section, we review the existing compression schemes that can be divided into redundancy-based, modular-based, and hash-based compression schemes. Those schemes commonly compress data strings to reduce the storage volume. In detail, redundancy-based compression eliminates redundant strings among data to replace them with numbers or

smaller strings. Modular-based compression reduces bits of data using a modular function to replace the data with smaller values. Those two types of compression can recover the original data by refilling or re-calculation, however, their compression efficiency is affected by redundancy and modulo of strings. Hash-based compression uses hash functions (e.g. SHA256) that compress data strings as a result of a fixed size. Although the one-way property of the hash functions does not allow the results to be recovered to the original data, the results can be used to verify the original. Moreover, hash-based compression can achieve a higher fixed compression efficiency because it generates results in a fixed size.

A. REDUNDANCY-BASED COMPRESSION

Traditionally, compression schemes that reduce the volume of data by eliminating and replacing the redundancy within the data strings have been used to solve the storage problem of traditional cryptocurrency systems. A compression scheme was proposed in [21] using the traditional compression techniques, LZ77 and Huffman tree, to achieve a higher storage efficiency. In this scheme, these techniques were used to compress a summary block proposed in [18] for the Bitcoin blockchain, and their smaller-sized results replaced the historical blocks. Thus, the scheme can achieve a higher storage efficiency than that of the previously proposed summarization scheme. In [27], a compression algorithm was proposed to reduce the size of a blockchain so that the nodes participating in a blockchain system could download the blockchain quickly. The algorithm compresses the sizes of transactions in the Bitcoin blockchain by replacing the existing hash pointers with index pointers of smaller sizes. Thus, the algorithm can reduce the transmission bandwidth for downloading the blockchain by creating a smaller-sized blockchain from which the original blockchain can be recovered. Moreover, a compression scheme was proposed to solve the storage problem of smart contracts in the Ethereum blockchain [22]. This scheme contains a new pseudo-opcode and a method to process the replacement of strings in new smart contracts. To reduce the storage volume of smart contracts, the scheme finds potential pointers that can connect from a newly generated contract to the existing deployed contracts using the proposed pseudo-opcode.

Redundancy-based compression schemes have the advantage of self-recovery by refilling the erased strings while achieving a reduction in storage volume. However, the existing schemes are not suitable for blockchain nodes with limited resources. These schemes have a limitation in that they cannot provide a fixed-size compression result because they depend on the redundancy of the data strings. Furthermore, they require a continuous recovery process to find and refill erased strings each time the original data are verified.

B. MODULAR-BASED COMPRESSION

A scheme for optimizing the storage mechanism based on the RNS was proposed in [23]. Based on the Chinese remainder theorem (CRT), which enables the RNS, a certain integer can be replaced by a smaller and recoverable integer through modular operations. In this scheme, each node randomly selects a modulo from a predefined modulo set when it participates in a blockchain system. The node only needs to maintain a smaller-sized remainder of the account information calculated by the selected modulo. The original can be recovered by collecting other remainders and modulos from other nodes. Moreover, the scheme can identify and recover intentional errors from devil nodes with low complexity of both the computation and communication, using redundant RNS and the new CRT. Thus, this scheme can effectively reduce the storage volume and also ensure security by detecting errors in the recovery process.

The scheme was originally designed to replace account information, which can be summarized and has a certain number of bits, with smaller remainder values. However, if the scheme is expanded to be applied to blockchain data instead of account information, the verification capabilities of nodes are hindered and the recovery process becomes inefficient. The compression results generated by applying the scheme to each blockchain datum eventually accumulate as the blockchain grows. Furthermore, even if the original copy exists, each of these remainders requires communication during the recovery process because they cannot be recovered without modulos from the other nodes. Although this scheme can achieve a consistently high storage efficiency compared with redundancy-based schemes, the data-dependent problem remains to be addressed to expand to the blockchain data.

C. HASH-BASED COMPRESSION

In [26], an efficient public blockchain client (EPBC) scheme was proposed to ensure not only the reduction of storage volume but also the verifiability of nodes in public blockchain systems. The scheme generates a summary of constant size by compressing the existing blockchain using a cryptography accumulator. A cryptography accumulator accumulates hash functions, and it is used to verify that each block exists inside the blockchain. In this scheme, nodes with limited resources receive and maintain the most up-to-date summary from the other nodes that are updated every time new blocks are generated. The nodes can perform their own verification by receiving the necessary proofs from the other nodes to verify which blocks belong to the blockchain. The inclusion of blocks is confirmed by performing a hash function and power calculations with the blocks and proofs. Although this scheme can achieve a high storage efficiency with verifiability, it has some limitations in terms of communication and verification. The nodes using this scheme must periodically receive and update the latest summary and receive proofs for each block every time the verification is performed. Furthermore, the proofs for the verification correspond to each block, hence, the number of proofs required is proportional to the number of blocks. The accumulation of proofs for verification can prevent dependent verification and frequent communication. However, the accumulation of proofs degrades the

verification capabilities of the nodes, so a way to prevent the accumulation should be considered.

Bitcoin includes a simplified payment verification (SPV) scheme for nodes with limited resources simply to engage in transactions [24]. For the nodes, it is difficult to perform the proof-of-work (PoW) consensus algorithm and maintain a blockchain completely. The SPV scheme allows the nodes to generate transactions and verify that the transactions have been processed. A block of the Bitcoin blockchain is divided into a block header and block body. The block header contains the root of the Merkle tree, called the Merkle root, which is obtained using the transactions contained in the block body. Nodes with limited resources only need to store a set of block headers from the longest Bitcoin blockchain. Then, the verification regarding whether a block contains a transaction requires the transaction, the Merkle root in the block header, and the Merkle paths to the Merkle root. The verification is performed by calculating the Merkle root through the transaction and Merkle paths and verifying that the calculated Merkle root is the same as the Merkle root in the block header that has been stored. Therefore, the SPV scheme can be used to solve the storage problem of nodes with limited resources in the Bitcoin blockchain. In addition, a compressible blockchain scheme was proposed by creating a second blockchain to compress the existing blockchain [20]. A second blockchain, called a snapshot chain, stores snapshots that have the same role as the blocks of the blockchain. A snapshot is generated every time a certain number of blocks are accumulated, and it contains the current UTXO set. The current UTXO set is constructed using the UTXO set in the previous snapshot and the transactions in the summarized blocks. The snapshots contain the summaries of transactions, so that the summarized blocks within the existing blocks can be erased. Thus, this scheme can be used to solve the storage problem because periodically created snapshots can replace the summarized blocks. In fact, the scheme was originally designed as a summarization scheme because a UTXO set can be constructed from the causality of a series of transactions. However, as mentioned in this scheme, saving a set of block headers instead of UTXO sets can function like the SPV scheme. This extended scheme can be considered to be a compression scheme that can provide a performance similar to that of the SPV scheme in that it uses a set of block headers.

However, a hash-based compression scheme using a traditionally used set of block headers has the disadvantage of accumulating compression results. In the SPV scheme, a block header is smaller than an entire block. However, the block headers are eventually accumulated proportionally to the number of blocks. Similarly, the snapshots are accumulated proportionally to the number of blocks in a compressible blockchain scheme. Therefore, both schemes still do not achieve a higher verification capability of nodes with limited resources due to the accumulation of compression results. The proposed SELCOM adopts the advantage of the existing hash-based compression schemes as it extends our previous research. In SELCOM, periodic checkpoint generation containing a block Merkle root (BMR) and update process can reduce the storage overhead while preventing the accumulation of compression results to achieve a higher verification capability on each node with limited resources. The proposed scheme belongs to hash-based compression as it uses a hash function to compress the cumulative blocks.

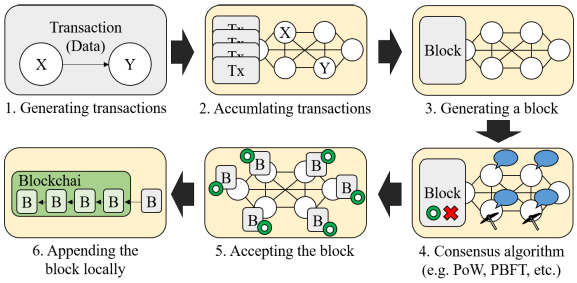


FIGURE 1. Workflow of traditional linear blockchain.

III. BACKGROUND

A. STORAGE FOR LINEAR BLOCKCHAIN

A linear blockchain is a type of blockchain with a ledger structure consisting of a series of blocks. It stores the blocks that have been processed sequentially through consensus algorithms. Traditional blockchain systems use the linear blockchain to deal with account information and balances, rather than real-world data. Figure 1 shows the workflow of the traditional linear blockchain. The workflow is described in detail as follows:

1) A new transaction is generated between blockchain nodes, and the node that receives the transaction broadcasts it to the blockchain network.
2) The received transaction is locally validated on each node. Then, the valid transactions are accumulated until they are sufficient to create a block.
3) A new block containing the accumulated transactions is created from a certain node (e.g. a miner or leader), and the node broadcasts it to the blockchain network.
4) During the consensus algorithm, the other nodes verify that the block is legitimate. In particular, the traditionally used PoW includes the preceding process of creating a block.
5) Based on the consensus algorithm, each node decides to accept the block when it is verified that the block is legitimate.
6) Finally, the block is appended to the blockchain stored locally on each node; consequently, the nodes are synchronized with the latest blockchain.

In a blockchain system, there are full or lightweight nodes depending on their capabilities, such as computation powers, batteries, and storage volumes. Full nodes are capable of handling a complete blockchain. In contrast, lightweight nodes are not sufficiently capable of handling the complete blockchain. Each node generates and verifies the transactions and blocks through a consensus algorithm to maintain the

linear blockchain. The blocks in the linear blockchain consist of finalized blocks and up-to-date blocks waiting for confirmation. If the nodes using the linear blockchain are synchronized, they have the same state of the distributed ledger by the finalized blocks. If the size of each block is S bytes and the total number of finalized blocks is n, the total size of the complete blockchain stored by each node is calculated as $n \times S$ bytes. If the storage volume threshold that a lightweight node can allow maintaining a blockchain is τ, the lightweight node suffers from the storage problem when $nS \geq \tau$. Then, the lightweight node can only perform verification on a series of n' blocks without the other nodes, where $n' \leq \lfloor \frac{\tau}{S} \rfloor$.

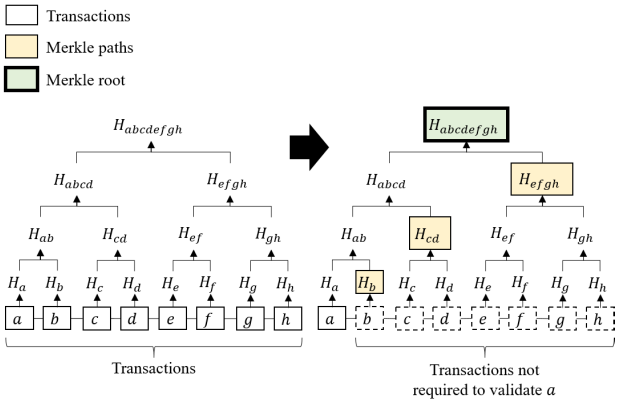


FIGURE 2. An example of Merkle tree by compressing 8 transactions.

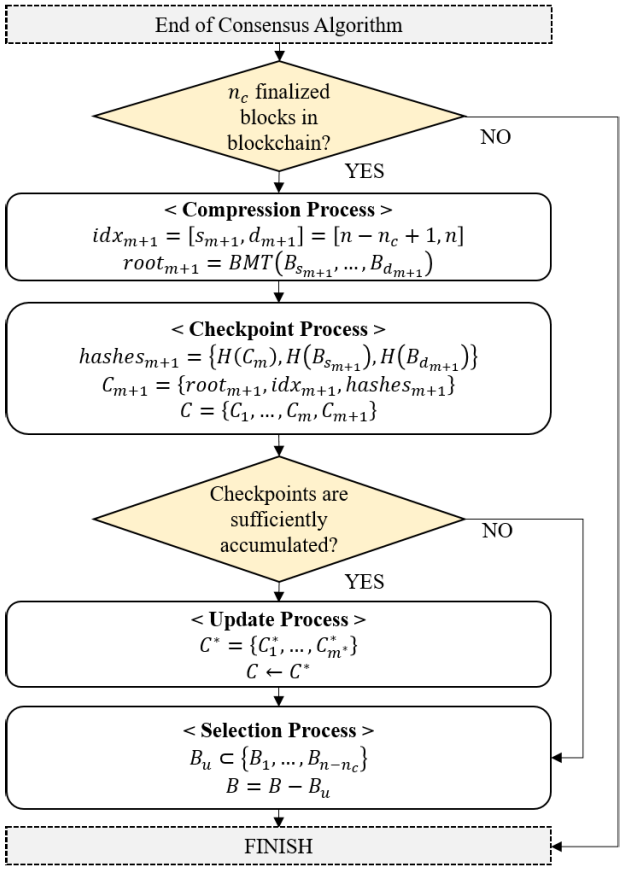


FIGURE 3. Sequential diagram of the proposed SELCOM.

B. MERKLE TREE

Merkle tree is a type of hash tree used to achieve efficient integrity verification in Bitcoin blockchain [24]. Figure 2 shows an example of a Merkle tree with integrity verification. H_{xy} is the hash result of the concatenated x and y entered. For example, $H_{ab} = H(H_a||H_b)$, where $H(\cdot)$ is a hash function used to compress an input. Several transactions are compressed into a single Merkle root through a series of repetitive hash functions. The calculated Merkle root is used to verify the compressed transactions with several Merkle paths. Suppose, as shown in the example, only a transaction a among the compressed transactions remains. The transactions a can still be verified by calculating the Merkle root with the Merkle paths, i.e., H_b, H_{cd}, and H_{efgh}. If the calculated Merkle root is the same as the Merkle root stored separately, the integrity of the transaction is ensured. Thus, the integrity of the compressed transactions can be independently verified through the Merkle tree, even if some transactions are erased.

IV. PROPOSED SCHEME

A. OVERALL PROCESSES

SELCOM is performed to reduce a storage volume for the blockchain, leaving only what is needed on each node for verification. It consists of four processes: compression, checkpoint, update, and selection. Unlike other processes, the update process is an additional process performed when the checkpoints are sufficiently accumulated to constitute the updated checkpoint.

Figure 3 shows a sequential diagram of the proposed SELCOM. In the compression process, newly accumulated n_c blocks are compressed into a new BMT. The BMT is used to generate a checkpoint in the checkpoint process. When checkpoints are sufficiently accumulated, the update process is performed to reduce the accumulation of checkpoints by combining them. Lastly, each node selects blocks to be left for independent verification and erases the rest to reduce the storage volume in the selection process. Note that we define the original linear blockchain as the main-chain and the proposed chain that stores checkpoints as the checkpoint-chain. In the overall processes, the main-chain and checkpoint-chain are described in detail as follows:

- $B = \{B_1, B_2, \cdots, B_n\}$; the main-chain set B contains finalized blocks B_i, where i is the block number. B does not contain a few up-to-date blocks waiting for confirmation. B_0 is the genesis block, but it is not compressed and is not included in B.
- $C = \{C_1, C_2, \cdots, C_m\}$; the checkpoint-chain set C contains checkpoints C_i, where i is the checkpoint number. If an update process has never been performed, $m = \left\lfloor \frac{n}{n_c} \right\rfloor$, as a checkpoint is newly generated with new n_c blocks in B.

- $C^* = \{C_1^*, C_2^*, \cdots, C_{m^*}^*\}$; the updated checkpoint-chain set C^* contains updated checkpoints C_i^*, where i is the updated checkpoint number. The update process combines the accumulated checkpoints into fewer checkpoints, hence, $m^* \leq m$.

B. SELECTIVE COMPRESSION SCHEME (SELCOM)

1) COMPRESSION

After the end of a consensus algorithm, n can be recalculated by updating the finalized blocks in the main-chain. A compression process is performed when $n_c|n$ with the recently accumulated n_c blocks, except for some of the latest blocks waiting for confirmation and historical blocks that have already been compressed into checkpoints. In the compression process, a BMT is used to compress the latest n_c blocks. The BMT is an extended version of the Merkle tree to compress several blocks in a blockchain instead of transactions. Thus, the compressed blocks can be independently verified with the BMR and BMPs. The index range idx_{m+1} of n_c uncompressed blocks at the rear end of B can be expressed as follows:

$$idx_{m+1} = [s_{m+1}, d_{m+1}] = [n - n_c + 1, n] \tag{1}$$

n_c blocks are compressed into a new BMT, hence, s_{m+1} and d_{m+1} are the block numbers of the leftmost and rightmost blocks in the BMT, respectively. Then, a new BMR $root_{m+1}$ is obtained as follows:

$$root_{m+1} = BMT(B_{s_{m+1}}, \cdots, B_{d_{m+1}}) \tag{2}$$

where $BMT(\cdot)$ is a function for building a BMT with input blocks, which returns the calculated BMR.

2) CHECKPOINT

In the checkpoint process, a new checkpoint is generated and stored in a checkpoint-chain. Checkpoints serve as compression results instead of historical blocks stored in a BMT and are connected to the checkpoint-chain, such as blocks in the main-chain. Figure 4 shows the overall structure of the proposed scheme, including the detailed structure of the blocks and checkpoints in the main-chain and checkpoint-chain, respectively. The blocks in the main-chain have the same structure as that of the traditional linear blockchain. In contrast, checkpoints contain various types of information to ensure the previous checkpoint and compressed blocks. Thus, with the accumulation of checkpoints, the main-chain in which the compressed blocks are stored can be more freely managed.

A new checkpoint C_{m+1} is generated and appended at the rear end of the latest checkpoint C_m. Note that if a checkpoint is first generated in a checkpoint-chain, the genesis block is regarded as the latest checkpoint that can be expressed as $C_0 = B_0$, but is not counted in C. A new checkpoint C_{m+1} contains the following information:

$$C_{m+1} = \{hashes_{m+1}, root_{m+1}, idx_{m+1}\} \tag{3}$$

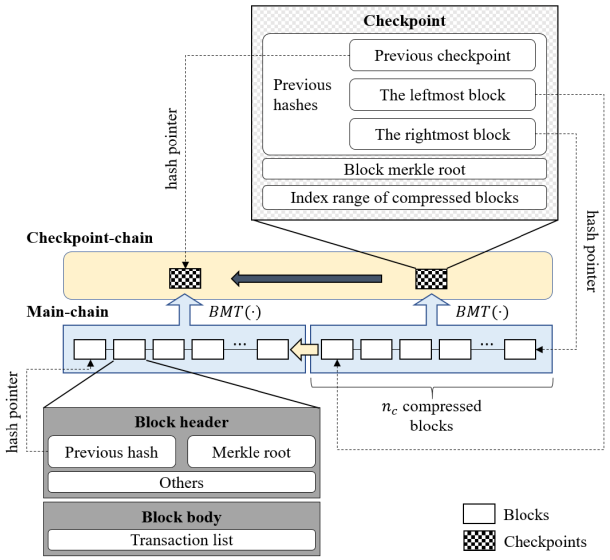


FIGURE 4. Overall structure of the proposed SELCOM scheme.

where $hashes_{m+1} = \{H(C_m), H(B_{s_{m+1}}), H(B_{d_{m+1}})\} = \{H_{m+1}^1, H_{m+1}^2, H_{m+1}^3\}$ is a set of previous hashes. The previous hashes contain the hash results of the previous checkpoint, and the leftmost and rightmost blocks among n_c compressed blocks. This information is used to check the scope of the compressed blocks and perform their verification. Thus, the checkpoint-chain is connected to the main-chain.

After the generation of C_{m+1}, the new checkpoint is appended at the rear end of the checkpoint-chain, which would have a length of $m + 1$. The integrity of compressed blocks can be verified using stored checkpoints. In summary, a checkpoint with a fixed size is efficient in terms of storage volume because it can replace n_c blocks. However, as with the existing schemes previously, the continuous accumulation of compression results may eventually be inefficient over time. Therefore, the checkpoint-chain is checked to perform the update process before proceeding to the selection process.

3) UPDATE

The update process combines the stored checkpoints in the checkpoint-chain to prevent their accumulation while preserving the purpose of the checkpoints. The update process is performed using Algorithm 1. Note that $(\cdot)^*$ is the updated information to be stored in an updated checkpoint. For an efficient verification through checkpoints, the property of a BMT, the balanced binary tree, should be preserved. The heights of the left and right subtrees of a balanced binary tree must be the same to increase its height. Thus, when the heights of the two latest subsequent checkpoints C_{m-1} and C_m are the same, an update process is performed. The heights can be obtained using idx_{m-1} and idx_m, respectively. In the update process, these checkpoints are combined into a single updated checkpoint. A new BMR $root^*$ is calculated

Algorithm 1 UPDATE: Update Process of Checkpoint-Chain

Input: C of length m
Output: C^* of length m^*

if $m \geq 2$ and $d_m - s_m == d_{m-1} - s_{m-1}$ **then**
  $root^* = H(root_{m-1}||root_m)$;
  $idx^* = [s_{m-1}, d_m]$;
  $hashes^* = \{H(C_{m-2}), H(B_{s_{m-1}}), H(B_{d_m})\}$;
  $C^*_{m^*} = \{hashes^*, root^*, idx^*\}$;
  $C^* = \{C_1, \ldots, C_{m-2}, C^*_{m^*}\}$
  return UPDATE(C^*)
else
  return C;
end if

Algorithm 2 FIRST-UPDATE: The First Update Process for a Checkpoint-Chain That Has Never Been Updated

Input: C of length m
Output: C^* of length m^*

$i = 2$;
$C_{temp} = \{C_1\}$;
while $i \leq m$ **do**
  $C_{temp} = C_{temp} \cup \{C_i\}$;
  $C_{temp} =$ UPDATE(C_{temp}) by Algorithm 1;
  $i = i + 1$;
end while
$C^* = C_{temp}$;
return C^*;

from the BMRs, $root_{m-1}$ and $root_m$, stored at each checkpoint. Furthermore, *idx* and *hashes* in these checkpoints are updated. Then, a newly updated checkpoint containing new information replaces the two checkpoints. This update process is repeated as much as possible on all the checkpoints from the rear end of the checkpoint-chain.

The update process may not be performed on each node because of its sufficient capabilities. The previous checkpoints may vary depending on whether an update process is performed. If the update process is performed once, the previous checkpoint would be the updated checkpoint. Then, H^1_{m+1} in a new checkpoint C_{m+1} can be different, hence, the recent blocks compressed into C_{m+1} are not confirmed. The newly generated checkpoint with its compressed blocks is confirmed if the next checkpoint is appended. Consequently, the checkpoint-chain with m checkpoints in the nodes can be synchronized if the next checkpoint C_{m+1} is appended or the update process is performed on its own. Algorithm 2 is used to perform an update process if a node that accumulates checkpoints without performing an update process wants to perform the process. Algorithm 1 is executed repeatedly from the initial to the latest generation of checkpoints. Then, a node can maintain the latest updated checkpoint-chain if Algorithm 2 is performed once.

As the update process preserves the property of a BMT, it may occur frequently in the early stages, but it shrinks over time. Consequently, this update process prevents the accumulation of compression results over time because it can compress numerous blocks with fewer checkpoints. We present the number of blocks that only a few updated checkpoints can cover with the update process in Section V.

4) SELECTION

In the selection process, the compressed blocks among the finalized blocks can be erased or retained based on the promise of the selectability of compressed blocks through checkpoints. Compressed blocks are divided into potential and useless blocks via the individual determination of each node. When each node decides to use some blocks later, the blocks are potential blocks within the node. The rest of the blocks are considered useless and would be erased. For example, a potential block contains data from other nodes that the node frequently communicates or trades with. In contrast, a useless block contains data that the node is not interested in. Each node calculates and maintains the BMPs to verify the integrity of the chosen potential blocks and then erases the useless blocks.

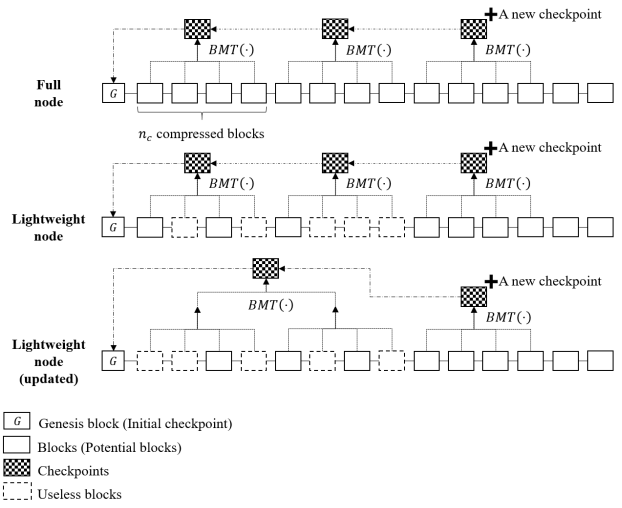


FIGURE 5. Example of SELCOM results in nodes after appending a new checkpoint, where $n_c = 4$.

Figure 5 shows an example of SELCOM in full and lightweight nodes after appending a new checkpoint. In the example, a checkpoint is generated every time $n_c = 4$ blocks are accumulated. Full nodes with sufficient capabilities generate checkpoints periodically but do not need to divide the blocks in the main-chain into potential and useless blocks. In contrast, lightweight nodes divide the blocks compressed in the previous checkpoints into potential and useless blocks as a new checkpoint is appended. Moreover, lightweight nodes proceed with the update process to prevent the accumulation of checkpoints. The lightweight node below has fewer checkpoints because it performed an update process when appending the previous checkpoint. Then, useless blocks B_u can be erased, where $B_u \subset \{B_1, \ldots, B_{n-n_c}\}$. The historical

blocks compressed into the previous checkpoints, but not the recent checkpoint, can be useless blocks. Through the selection process, there is an update of the main-chain $B = B - B_u$ inside the node with only potential blocks remaining.

SELCOM ensures flexible data (block) management from the selectability of compressed blocks in the main-chain and solves the storage problem. The smaller the number of potential blocks that remain, the lower is the storage volume of the main-chain. In contrast, each node can flexibly determine to remain the maximum number of potential blocks based on its own capabilities to achieve a higher verification capability. In particular, while the existing schemes accumulate compression results, SELCOM can prevent accumulation through the update process. Thus, SELCOM also ensures a higher verification capability under the same storage volume of any node because the prevention reduces the storage overhead. Meanwhile, SELCOM should manage BMPs for verification of potential blocks instead of accumulating compression results. Nevertheless, it is confirmed in the following section that SELCOM achieves a higher storage efficiency compared with the existing schemes.

V. MATHEMATICAL ANALYSIS AND RESULTS

In this section, the proposed and existing schemes are compared in terms of storage efficiency and verification capability by the numerical results based on the analyses. The notations and their descriptions used in this section are listed in Table 1.

TABLE 1. Notations with descriptions for analysis.

Notation	Description	Domain
n	Number of finalized blocks	$n \in \mathbb{N}$
m	Number of checkpoints	$m \in \mathbb{N}$
m^*	Number of updated checkpoints	$m^* \in \mathbb{N}$
n_c	Number of compressed blocks	$n_c \vert n, n_c \in \mathbb{N}$
bmp	Number of block Merkle paths	$bmp \in \mathbb{N}$
r	Number of potential blocks	$r \in [0, n], r \in \mathbb{N}$
S_{hash}	Size of a hash result value	In bytes
S_{header}	Size of a block header	In bytes
S	Size of a block	In bytes
V	Storage volume of compression results	In bytes

Note that $n_c = 2^x$, where $x \geq 1$ and $x \in \mathbb{N}$, to construct a BMT that is a balanced binary tree. By default, x is a constant value, however, it can be adjusted through the consensus algorithm for the blockchain system for a higher reduction in storage. The update process combines checkpoints with the same x height. Then, the updated checkpoints have different heights that are larger than x, and the latest checkpoint has the lowest height. Thus, x can be adjusted to a value less than or equal to the lowest height to still preserve the property of a balanced binary tree.

In SELCOM, nodes leave r potential blocks based on the selectability of blocks in the main-chain. The residual r blocks improve the verification capabilities of the nodes, although they increase the storage volume. The nodes should store additional BMPs for the internal verification of the r blocks. Thus, r blocks dispersed across m checkpoints can be expressed as follows:

$$r = \sum_{i=1}^{m} r_i \tag{4}$$

where $0 \leq r_i \leq n$ and r_i is the number of potential blocks compressed into a checkpoint C_i. Then, the number of BMPs bmp depends on r_i from the BMT of each checkpoint. If these r blocks are dispersed at only a few checkpoints, bmp becomes smaller because these blocks can be used to calculate the BMPs. Furthermore, if the potential blocks are concentrated to the left or right at the BMT of a single checkpoint, bmp becomes smaller because the height of the opposite subtree can increase. Thus, we consider the worst-case scenario in which r blocks are equally dispersed not only across all the BMTs but also within each BMT, to check the performance of SELCOM in the worst case.

A. MATHEMATICAL ANALYSIS

The original size of a complete blockchain with finalized blocks can be expressed as nS bytes. The required storage volume of each scheme depends on the compressed results represented as follows:

1) EPBC scheme [26]: Lightweight nodes should receive and store the up-to-date summary from the other nodes to verify blocks. The summary is a hash result obtained using a hash accumulator, hence, it has a fixed size. The summary S_n is expressed as follows:

$$S_n = g^{\prod_{i=1}^{n} H(blk_i, i)} \mod N \tag{5}$$

where g is a random value, blk_i is a block with the number i, and N is the modulus for the RSA accumulator system. Then, S_n is the compression result of a complete blockchain in this scheme. In the verification of this scheme, the proof from other nodes is required for each block. The received proof $p_i = (p_i^{(1)}, p_i^{(2)})$ for a block blk_i is expressed as follows:

$$p_i = \begin{cases} p_i^{(1)} = H(blk_i, i), \\ p_i^{(2)} = g^{\prod_{k=1}^{n} H(blk_k, k)/H(blk_i, i)} \end{cases} \mod N. \tag{6}$$

where $p_i^{(1)}$ is a hash result from blk_i and $p_i^{(2)}$ is a hash accumulator calculated excluding $p_i^{(1)}$. Even in this scheme, blocks can be left as potential blocks to achieve the high verification capability of lightweight blocks. If blk_i is a potential block, $p_i^{(1)}$ can be calculated. On the other hand, $p^{(2)}$ is required for each potential block. Note that N is the product of two large prime numbers that are sufficiently large to be not easily found to ensure the security of this scheme based on RSA. Thus, the storage volume of the EPBC scheme V_{EPBC} with r potential blocks can be expressed as follows:

$$V_{EPBC} = |S_n| + r(S + |p^{(2)}|) \tag{7}$$

where $|S_n|$ and $|p^{(2)}|$ are the byte sizes of S_n and $p^{(2)}$, respectively. Note that S_n and $p^{(2)}$ are calculated as hash accumulators, and they are always smaller than N by a modular operation. We assume that these two values are $\frac{N}{2}$ on average, with the assumption that the result of $H(\cdot)$ used is uniformly random.

2) Snapshot scheme [20]: This scheme was originally designed to store a UTXO set in a snapshot which is a block of the second blockchain. Instead of a UTXO set, a set of headers can be stored in a snapshot to verify the compressed blocks by guaranteeing the selectability of the blocks. Thus, the compressed blocks can be potential blocks, hence, the storage volume of this scheme $V_{SNAPSHOT}$ can be expressed as follows:

$$V_{SNAPSHOT} = m(2 \times S_{hash} + n_c \times S_{header}) + r \times S \tag{8}$$

Note that m and n_c can be re-used in this scheme because the scheme also generates snapshots periodically with several blocks. A snapshot contains two hashes corresponding to the previous block in the original chain and the previous snapshot in the second blockchain.

3) SELCOM: The storage volume of the proposed scheme V_{SELCOM} that remains checkpoints, potential blocks, and BMPs for the verification of the blocks can be expressed as follows:

$$V_{SELCOM} = m(4 \times S_{hash} + 2 \times I) + \sum_{i=1}^{m}(r_i \times S + bmp_{x,r_i} \times S_{hash}) \tag{9}$$

where I is the integer size for the index range idx, and bmp_{x,r_i} is the number of BMPs for r_i blocks among $n_c = 2^x$ compressed blocks in a checkpoint C_i. If $r_i = 0$ within C_i, then $bmp_{x,r_i} = 0$ because no BMPs are required. Moreover, if one potential block remains within one BMT ($r_i = 1$), then $bmp_{x,r_i} = x$ with a BMR already stored in a checkpoint. In contrast, if multiple blocks remain within one BMT, the BMPs can overlap for each block. Then, it is better to store the hash results of sibling blocks (useless blocks) instead of the BMPs. Thus, bmp_{x,r_i} is the number of BMPs from each potential block to the root of the subtree with each height of the BMTs, and it is expressed as follows:

$$bmp_{x,r_i} = \begin{cases} 0, & \text{if } r_i = 0; \\ r_i(x - k_i - 1) + 2^{k_i}, & \text{if } r_i \le \dfrac{n_c}{2}; \\ n_c - r_i, & \text{otherwise.} \end{cases} \tag{10}$$

where $k_i = \lceil \log_2 r_i \rceil$. If more potential blocks than half of the compressed n_c blocks remain in a BMT, storing the hash results of sibling blocks is efficient in terms of storage volume.

4) SELCOM (updated): An update process combines as many checkpoints as possible to reduce the number of checkpoints. The update process is repeated until all the updated checkpoints have BMTs with different heights. Then, the checkpoints are no longer updated until the next checkpoint is appended. The storage volume of SELCOM with an update process V_{SELCOM^*} can be expressed based on (9) as follows:

$$V_{SELCOM^*} = m^*(4 \times S_{hash} + 2 \times I) + \sum_{j=1}^{w}(r_j^* \times S + bmp_{y_j,r_j^*} \times S_{hash}) \tag{11}$$

The updated checkpoints have different heights of the BMTs. Moreover, as BMTs are combined in the update process, the older updated checkpoint has a BMT with a greater height. Thus, the updated checkpoints can be generated depending on whether a BMT with a certain height can be generated. Then, the number of updated checkpoints m^* can be obtained as follows:

$$m^* = \sum_{i=j}^{w} b_j \tag{12}$$

where b_j is a binary value that indicates the existence of an updated checkpoint that can be constructed as best as possible from m checkpoints, and w is the maximum height of a BMT that can be constructed by m. Then, $w = \lfloor \log_2 m \rfloor + 1$ and $1 \le j \le w$. Intuitively, if a BMT with height $j-1$ can be constructed, an updated checkpoint exists with $b_j = 1$. Thus, a set of b_j is obtained by satisfying $\sum_{j=1}^{w} b_j \times 2^{j-1} = m$, where $b_j \in \{0, 1\}$ (In fact, the set of b_j is the same as the result of reversing the binary number of n_c). Finally, bmp_{y_j,r_j^*} is the number of BMPs for r_j^* potential blocks among 2^{y_j} compressed blocks, where $y_j = x + j - 1$. r_j^* is the total number of potential blocks in an updated checkpoint if it exists. Then, depending on the range of the combined checkpoints, r_j^* can be expressed as follows:

$$r_j^* = \begin{cases} 0, & \text{if } b_j = 0; \\ \sum_{i=s_j}^{d_j} r_i, & \text{otherwise.} \end{cases} \tag{13}$$

where $s_j = m - \sum_{l=1}^{j} b_l \times 2^{l-1} + 1$ and $d_j = m - \sum_{l=1}^{j} b_l \times 2^{l-1} + b_j \times 2^{j-1}$. The range $[s_j, d_j]$ indicates the number range of existing checkpoints combined in an updated checkpoint. Moreover, the number of compressed blocks is $n_c \times 2^{j-1}$ in an updated checkpoint with $b_j = 1$, hence, r_j^* has the range of $0 \le r_j^* \le n \times 2^{j-1}$. Consequently, bmp_{y_j,r_j^*} based on (10) can be obtained as follows:

$$bmp_{y_j,r_j^*} = \begin{cases} 0, & \text{if } r_j^* = 0; \\ r_j^*(y_j - k_j^* - 1) + 2^{k_j^*}, & \text{if } r_j^* \le \dfrac{2^{y_j}}{2}; \\ 2^{y_j} - r_j^*, & \text{otherwise.} \end{cases} \tag{14}$$

where $k_j^* = \lceil \log_2 r_j^* \rceil$. In conclusion, we obtain the storage volume of the compressed results necessary to calculate the storage efficiency of the existing and proposed schemes.

The following is an example of calculating the storage volume of the compression results in the proposed scheme. Assume that there are 320 blocks in a main-chain $n = 320$, and a checkpoint is newly generated every 16 blocks $n_c = 16$. Then, there are $m = 20$ checkpoints stored in a checkpoint-chain without any update process. After an update process, the checkpoints are replaced with only two updated checkpoints calculated by $20 = 0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 = 10100_{(2)}$ with $w = 5$. Then, $b = \{b_1, b_2, b_3, b_4, b_5\} = \{0, 0, 1, 0, 1\}$. The two updated checkpoints are constructed using 4 checkpoints ($b_3 = 1$) and 16 checkpoints ($b_5 = 1$), respectively. In the worst case, the potential blocks are equally dispersed in each checkpoint. Assume that $r = 140$ among $n = 320$; then, $r_i = \frac{r}{m} = \frac{140}{20} = 7$. By (13), $r_3^* = \sum_{s_3}^{d_3} r_i = \sum_{17}^{20} 7 = 28$ and $r_5^* = \sum_{s_5}^{d_5} r_i = \sum_{1}^{16} 7 = 112$. Finally, $bmp_{y_3, r_3^*} + bmp_{y_5, r_5^*} = 160$ hashes are required to verify 140 potential blocks as BMPs with updated checkpoints.

Over time, the existing and proposed schemes accumulate compression results or proofs for verification. These accumulations increase the storage volume on each node, thus hindering the verification capability of each node. However, the proposed scheme with an update process can achieve a higher verification capability because m^* updated checkpoints can cover numerous blocks through the structural characteristics of BMTs. If all b_j calculated from m are 1, $m^* = w$, which is the largest value, given m, according to (12). Among m^* updated checkpoints, the oldest has the BMT compressing $n_c \times 2^{w-1}$ blocks, and sequentially, the most recent one has the BMT compressing n_c blocks. Then, the total number of blocks covered by m^* updated checkpoints can be calculated as $\sum_{i=1}^{m^*} n_c \times 2^{i-1} = n_c(2^{m^*} - 1)$. In contrast, m snapshots can cover $m \times n_c$ blocks in the snapshot scheme because the scheme generates a snapshot every n_c blocks. For example, assume that there are 2,096,128 blocks, which is three times the number of blocks in the current Bitcoin blockchain. These blocks are compressed into 2047 checkpoints (or snapshots) if $n_c = 1024$. After an update process, these 2047 checkpoints are replaced with only 11 updated checkpoints. Thus, in terms of storage efficiency, the proposed scheme is more efficient than the existing schemes, because it can be used to verify numerous blocks with fewer compression results.

B. NUMERICAL RESULTS

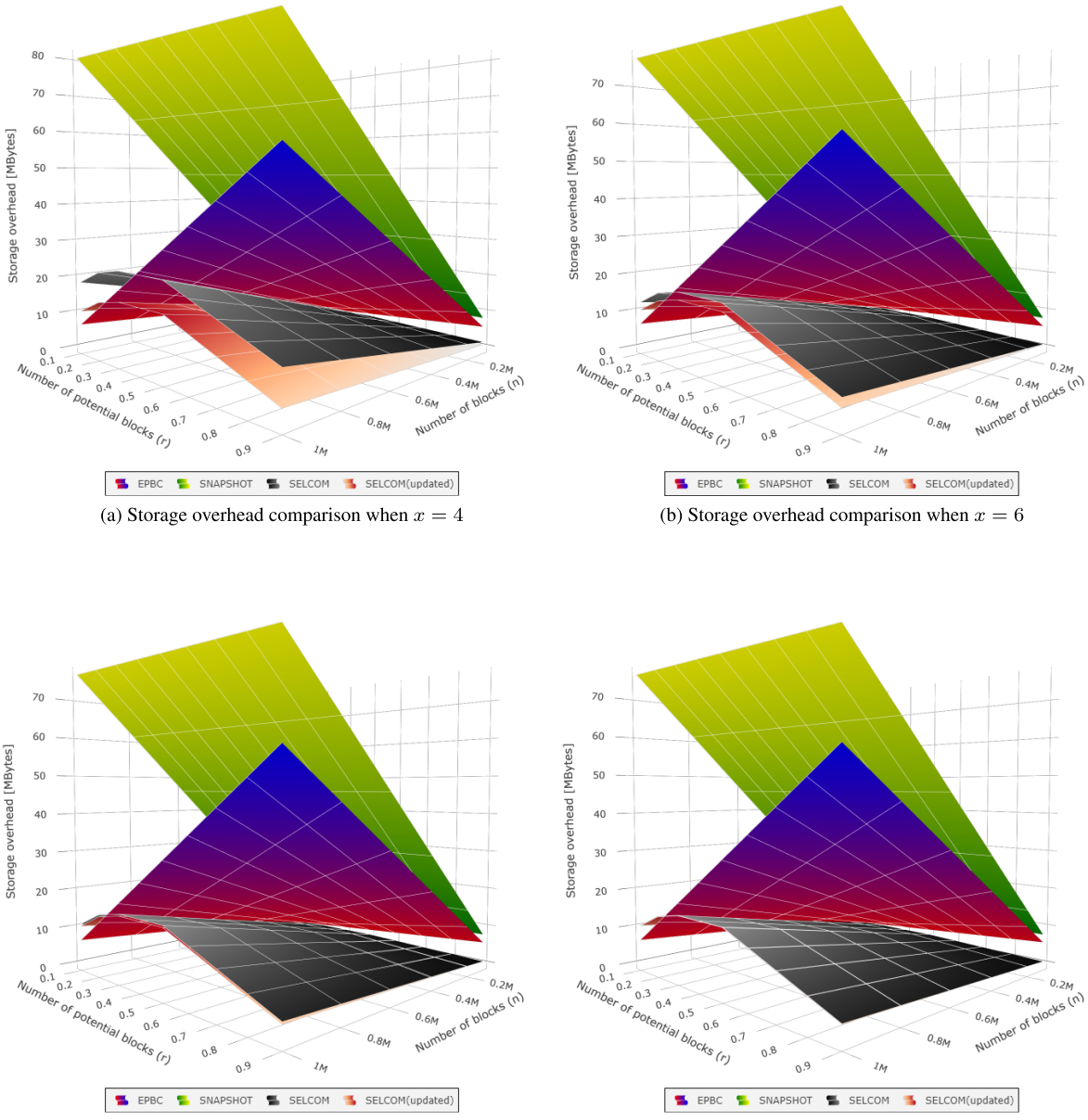
Numerical results were obtained by calculating the storage volumes of the analyzed schemes using the parameters in Table 2. The storage overheads of the schemes were calculated to show that the proposed scheme prevents the accumulation of compression results. Moreover, the number of blocks that can be maintained to achieve a higher verification capability under the same storage volume of a node was calculated. In conclusion, the proposed scheme shows that verification through potential blocks can be ensured without the accumulation of compression results. Note that the proposed and snapshot schemes construct a second blockchain, in contrast to the EPBC scheme.

TABLE 2. Parameters for numerical comparisons.

Notation	Description	Used parameter
n	Number of finalized blocks	$[100K, 1M]$
n_c	Number of compressed blocks	$[2^4, 2^{10}]$
r	Number of potential blocks	$[0.1, 0.9] \times n$
S_{hash}	Size of a hash result value	32 bytes
S_{header}	Size of a block header	80 bytes [24]
S	Size of a block	1 Mbytes
N	Modulus for RSA assumption	128 bytes [26]

Figure 6 shows the storage overhead comparison of the analyzed schemes. The x-axis is the number of finalized blocks n stored in a complete blockchain, and the y-axis is the number of potential blocks r proportional to n. The storage overhead, which is the z-axis, is the calculated result from the compression results for a given n and r, excluding the storage volume to maintain the stored blocks. Each subfigure represents the result according to x calculated from n_c, the number of compressed blocks. As the z-axis is the storage overhead, the lower the surface, the more efficient the scheme is, in terms of storage. In general, as n and r increase, the compression results accumulate, resulting in a higher storage overhead. In summary, the existing schemes with cumulative compression results require a higher storage overhead, as the number of blocks to be verified increases over time with the growth of the blockchain. The compression results in the EPBC scheme are not affected by n, but the storage overhead increases linearly as r increases. Moreover, the compression results are not affected by x because this scheme does not use a second blockchain. Here, r is proportional to n, hence, the storage overhead of this scheme increases as n and r increase. In contrast, the compression results in the snapshot scheme are affected by n and x. As the snapshot, which are the compression results, store a set of headers, their storage overhead increases as n increases. Although this scheme may reduce the increasing rate of storage overhead as x increases through the snapshot blockchain, the storage overhead still increases linearly proportional to n.

Unlike in the existing schemes, the storage overhead of the proposed scheme increases by a low margin as n increases, moreover, if both n and r increase, the storage overhead decreases. The proposed scheme constructs a second blockchain, similar to the snapshot scheme. The checkpoints compress blocks through BMTs, and they store the BMRs calculated from the BMTs, hence, they are less affected by n than the snapshots. However, maintaining the proposed scheme requires BMPs, which affect the storage overhead. Nevertheless, because of the binary property of the tree structure of BMTs, the number of required BMPs decreases as r increases by more than half in the BMPs. Furthermore, the proposed scheme with an update process combines



(a) Storage overhead comparison when $x = 4$

(b) Storage overhead comparison when $x = 6$

(c) Storage overhead comparison when $x = 8$

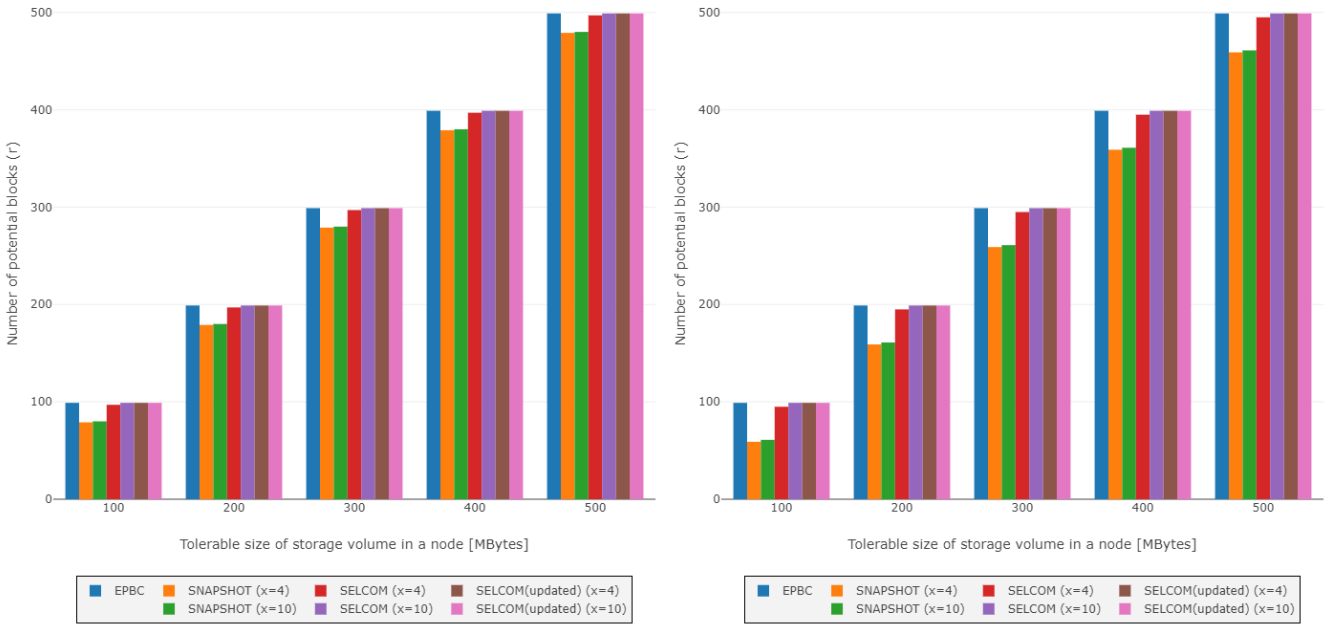
(d) Storage overhead comparison when $x = 10$

FIGURE 6. **Storage overhead comparison by number of blocks (*n*) and potential blocks (*r*).**

several accumulated checkpoints. When x increases, m becomes smaller for a given n, hence, the efficiency of the update process decreases. Nevertheless, an update process can be used efficiently to reduce the accumulated checkpoints because blocks continue to be created over time.

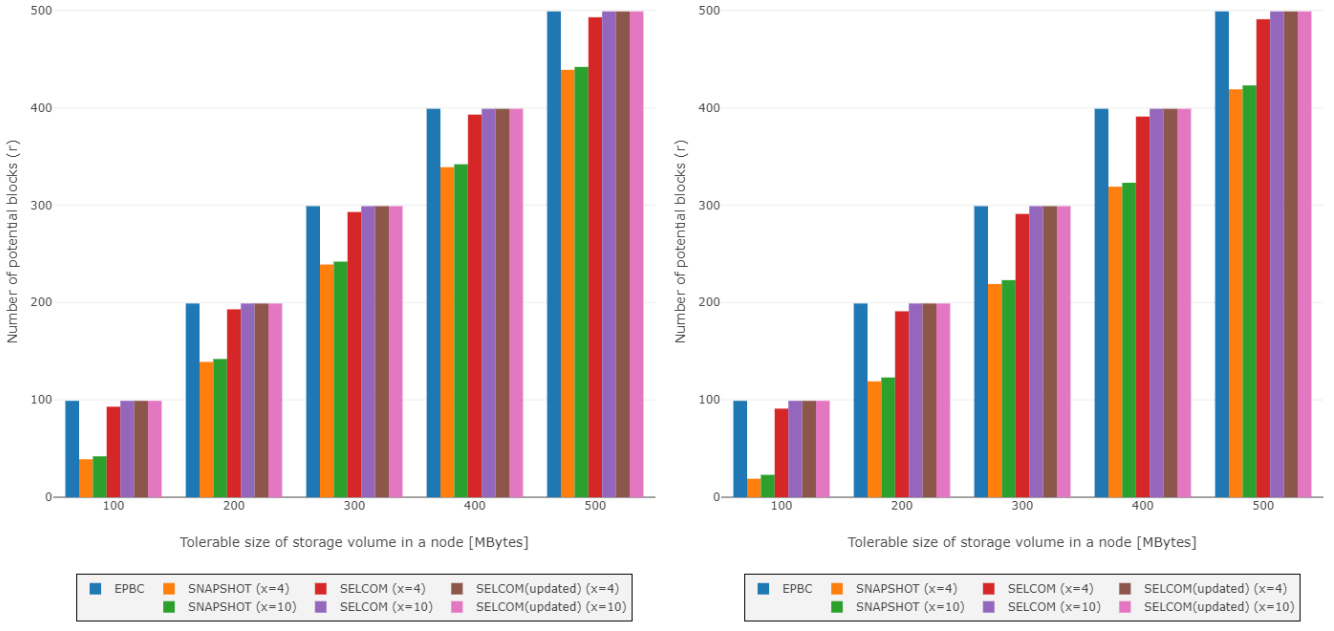
In conclusion, the proposed scheme is efficient in terms of storage because it can achieve a low level of storage overhead despite the increase in n and r. Specifically, in the comparison for $x = 4$, the proposed scheme achieved average reductions of 37.05% and 76.02% in the storage overhead compared with the EPBC and snapshot schemes, respectively. Furthermore, it achieved average reductions of 63.61% and 86.14%, respectively, in the storage overhead when accompanied by an update process.

Figure 7 shows the verification capability of the analyzed schemes under the same storage volume of a node. The x-axis is the storage volume allowed by each node for maintaining blocks, and the y-axis is the number of potential blocks r that can be maintained by each scheme with compression results. Each sub-figure is represented by the number of blocks n in a given blockchain. r indicates the number of tolerable blocks because it is the number of blocks remaining.



(a) Number of potential blocks when $n = 25K$

(b) Number of potential blocks when $n = 50K$



(c) Number of potential blocks when $n = 75K$

(d) Number of potential blocks when $n = 1M$

FIGURE 7. **Number of potential blocks (r) that can be maintained in a given blockchain by the tolerable size of storage volume (τ) in a node.**

The nodes can achieve a higher verification capability because the more blocks they can maintain, the more blocks they can verify. The traditional linear blockchain can verify a series of blocks, however, extracting arbitrary blocks is difficult. In contrast, the existing and proposed schemes can be used to maintain and validate blocks selectively.

It is apparent that r increases as the allowed storage volume increases. The EPBC scheme and proposed schemes achieve verification capabilities proportional to the tolerable storage volumes. While the EPBC scheme is not affected by n, the proposed scheme is slightly affected by n. Nevertheless, the proposed scheme achieves a similar level of

verification capabilities as the EPBC scheme when an update process is used. In particular, as these schemes can maintain blocks selectively proportional to the storage volumes as potential blocks, the storage overhead of these schemes is relatively low. Although higher verification capabilities are achieved, the EPBC scheme has a limitation in that it must update the compression results for each addition of a new block. The proofs for verification of potential blocks should be updated according to the latest summary received from the other nodes. In contrast, the proposed scheme maintains the checkpoint-chain but does not require checkpoints to be changed every time a new block is appended.

The snapshot scheme is affected by n because it constructs a second blockchain, similar to the proposed scheme. Then, as n increases, the storage overhead of the scheme increases, which decreases the verification capabilities of the scheme. The main difference between the snapshot and proposed schemes is the update process. As both schemes construct a second blockchain, the compression results inevitably accumulate in proportion to n, even if their cumulative sizes are different. However, the proposed scheme with an update process can reduce the accumulated compression results, hence, it can achieve a higher verification capability. As an example in figure 7b, when $x = 4$ in the proposed and snapshot schemes, the proposed scheme achieved an average increase of 13.90% in verification capability compared with the snapshot scheme. The corresponding average increase was 15.44% when using an update process. In conclusion, the proposed scheme can efficiently achieve higher verification capability under the same storage volume through a lower storage overhead.

VI. CONCLUSION

We proposed the SELCOM scheme to solve the storage problem without accumulations. The proposed scheme constructs the checkpoint-chain, which is a second blockchain, by extending our previous study with the concept of a BMT. A checkpoint is generated by compressed blocks in a blockchain. Checkpoints can be accumulated but this accumulation is resolved through the proposed update process. We mathematically analyzed the existing compression schemes and the proposed method. Moreover, we presented the numerical results calculated using the analysis and the parameters used in the traditional linear blockchain. In conclusion, the proposed scheme achieved higher storage efficiency and verification capabilities compared with the existing schemes. In particular, the proposed scheme can be expected to be more efficient as the size of a complete blockchain and the number of maintained blocks increase over time.

Nevertheless, the proposed scheme maintains compression results in the form of a second blockchain, requiring communication to perform real-time synchronization between nodes. In future works, the proposed scheme will be improved for an efficient sharing of compression results for checkpoint-chain synchronization and block verification when nodes with diverse resources organize systems hierarchically.

REFERENCES

[1] M. Attaran and A. Gunasekaran, "Blockchain-enabled technology: The emerging technology set to reshape and decentralise many industries," *Int. J. Appl. Decis. Sci.*, vol. 12, no. 4, pp. 424–444, 2019.

[2] E. Tijan, S. Aksentijević, K. Ivanić, and M. Jardas, "Blockchain technology implementation in logistics," *Sustainability*, vol. 11, no. 4, p. 1185, Feb. 2019.

[3] M. Pournader, Y. Shi, S. Seuring, and S. C. L. Koh, "Blockchain applications in supply chains, transport and logistics: A systematic review of the literature," *Int. J. Prod. Res.*, vol. 58, no. 7, pp. 2063–2081, Apr. 2020.

[4] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu, "The blockchain-based digital content distribution system," in *Proc. IEEE 5th Int. Conf. Big Data Cloud Comput.*, Aug. 2015, pp. 187–190.

[5] D. Bumblauskas, A. Mann, B. Dugan, and J. Rittmer, "A blockchain use case in food distribution: Do you know where your food has been?" *Int. J. Inf. Manage.*, vol. 52, Jun. 2020, Art. no. 102008.

[6] G. Song, S. Kim, H. Hwang, and K. Lee, "Blockchain-based notarization for social media," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–2.

[7] V. L. Lemieux, "Blockchain and distributed ledgers as trusted recordkeeping systems," in *Proc. Future Technol. Conf. (FTC)*, 2017, pp. 1–12.

[8] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.

[9] S. Jiang, H. Wu, and L. Wang, "Patients-controlled secure and privacy-preserving EHRs sharing scheme based on consortium blockchain," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[10] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Netw.*, vol. 33, no. 5, pp. 166–173, Sep. 2019.

[11] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207.

[12] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, p. 16440–16455, 2020.

[13] Blockchair. (Aug. 2020). *Blockchain Size Chart*. [Online]. Available: https://blockchair.com/ethereum/charts/blockchain-size?compare=bitcoin

[14] A. Shahnaz, U. Qamar, and A. Khalid, "Using blockchain for electronic health records," *IEEE Access*, vol. 7, pp. 147782–147795, 2019.

[15] J. Al-Jaroodi and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.

[16] S. Ghimire, J. Y. Choi, and B. Lee, "Using blockchain for improved video integrity verification," *IEEE Trans. Multimedia*, vol. 22, no. 1, pp. 108–121, Jan. 2020.

[17] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.

[18] A. Palai, M. Vora, and A. Shah, "Empowering light nodes in blockchains with block summarization," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5.

[19] J. Bruce, "The mini-blockchain scheme," White Paper, 2014.

[20] A. Marsalek, T. Zefferer, E. Fasllija, and D. Ziegler, "Tackling data inefficiency: Compressing the bitcoin blockchain," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 626–633.

[21] U. Nadiya, K. Mutijarsa, and C. Y. Rizqi, "Block summarization and compression in bitcoin blockchain," in *Proc. Int. Symp. Electron. Smart Devices (ISESD)*, Oct. 2018, pp. 1–4.

[22] B. B. Fiz Pontiveros, R. Norvill, and R. State, "Recycling smart contracts: Compression of the ethereum blockchain," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5.

[23] H. Mei, Z. Gao, Z. Guo, M. Zhao, and J. Yang, "Storage mechanism optimization in blockchain system based on residual number system," *IEEE Access*, vol. 7, pp. 114539–114546, 2019.

[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[25] T. Kim, J. Noh, and S. Cho, "SCC: Storage compression consensus for blockchain in lightweight IoT network," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–4.

[26] L. Xu, L. Chen, Z. Gao, S. Xu, and W. Shi, "Epbc: Efficient public blockchain client for lightweight users," in *Proc. 1st Workshop Scalable Resilient Infrastruct. Distrib. Ledgers*, 2017, pp. 1–6.
[27] X. Chen, S. Lin, and N. Yu, "Bitcoin blockchain compression algorithm for blank node synchronization," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2019, pp. 1–6.

TEASUNG KIM received the B.E. degree in computer science and engineering from Hanyang University, South Korea, in 2017, where he is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering. Since 2017, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include blockchain technology for lightweight systems, IoT security, and wireless communication systems.

SEJONG LEE received the B.E. degree in computer science and engineering from Jeju National University, South Korea, in 2018. He is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering with Hanyang University, South Korea. Since 2018, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include security for wireless communications, IoT security, and blockchain-based the medical data sharing systems.

YONGSEOK KWON received the B.E. degree in computer science and engineering from Hanyang University, South Korea, in 2019, where he is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering. Since 2019, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include blockchain security and applied blockchain, and smart contract.

JAEWON NOH received the B.E. degree in computer science and engineering from Hanyang University, South Korea, in 2015, where he is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering. Since 2015, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include security for wireless communication, wireless communication, and vehicular communication systems.

SOOHYEONG KIM received the B.E. degree in computer science and engineering from Hanyang University, South Korea, in 2019, where he is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering. Since 2019, he has been with the Computer Science and Engineering, South Korea. His research interests include blockchain technology, wireless communication, and applied blockchain to cellular communication.

SUNGHYUN CHO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Hanyang University, South Korea, in 1995, 1997, and 2001, respectively. From 2001 to 2006, he was with the Samsung Advanced Institute of Technology, and with the Telecommunication R&D Center of Samsung Electronics, where he has been engaged in the design and standardization of MAC and network layers of WiBro/WiMAX and 4G-LTE systems. From 2006 to 2008, he was a Postdoctoral Visiting Scholar with the Department of Electrical Engineering, Stanford University. He is currently a Professor with the Department of Computer Science and Engineering, Hanyang University. His primary research interests are 5th generation mobile communications, software defined networks, and vehicular communication systems. He is a member of the board of directors of the Institute of Electronics and Information Engineers (IEIE) and the Korean Institute of Communication Sciences (KICS).