Computers and Electrical Engineering 116 (2024) 109216

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Techniques to alleviate blockchain bloat: Potentials, challenges, and recommendations

Yehia Ibrahim Alzoubi[a], Alok Mishra[b,*]

[a] College of Business Administration, American University of the Middle East, Kuwait
[b] Faculty of Engineering, Norwegian University of Science and Technology (NTNU), Norway

ARTICLE INFO

Keywords:
Blockchain
Bloat
Security
Privacy
Internet of things (IoT)

ABSTRACT

Blockchain (BC) bloat, characterized by excessive data growth on BC networks, presents a pressing challenge. The need for a comprehensive study addressing BC bloat solutions arises due to the scarcity of knowledge in this emerging area. This study advances knowledge along with practical insights and inspires innovative approaches to address BC bloat. This research explores various techniques proposed to combat BC bloat. Two technique themes are identified: on-chain and off-chain techniques. On-chain techniques focus on updating the consensus mechanism, block size, or database partition. This study identified five on-chain techniques: structural modification, pruning, sharding, ephemeral BC, and zk-SNARKs. Off-chain techniques focus on processing in parallel, bundling transactions, or handling transactions off-chain. Further, three off-chain techniques identified are off-chain storage, historical data storage, and light clients' techniques. These recommendations aim to optimize resources use, enhance scalability, and maintain data integrity. However, these may compromise several BC capabilities, such as security, privacy, and data loss. The applicability of these techniques for resource-constrained devices is also explored. Future research should include further investigation into the performance and applicability of these techniques, especially in resource-constrained environments like IoT and fog computing.

1. Introduction

Although BC technology is not yet common, many more firms understand its usefulness. Businesses have employed BC technology to assist them in speeding up digital operations. The Statista Research Department expects an enormous increase in worldwide BC technology sales in the next few years, according to research released in September 2023, with the industry predicted to expand from 23 billion in 2023 to over 39 billion US dollars by 2025 [1]. A BC is a distributed system of nodes that independently processes cryptocurrency transactions and uses consensus or common rules to guarantee the accuracy of the transactions. Then, transactions are stored in consecutive sequences, producing a chain of immutable data blocks. As a result, what is known as ``BC bloat'' occurred. The continuous increase in a BC's size as more transactions and data are added to it is referred to as BC bloat, BC size, or BC growth. BC technology, the foundation of digital currencies like Bitcoin, consists of a sequence of blocks, each containing a collection of transactions and additional information. As more and more transactions get logged onto the BC, it steadily bulges in size, much like a well-stuffed filing cabinet [2]. To keep things running smoothly and efficiently in the face of this ever-growing network burden, future

* Corresponding author.
E-mail address: alok.mishra@ntnu.no (A. Mishra).

https://doi.org/10.1016/j.compeleceng.2024.109216
Received 10 December 2023; Received in revised form 16 March 2024; Accepted 21 March 2024

BC networks will likely lean heavily on innovative solutions to manage this data bloat. In the future, BC networks will depend more and more on bloat solutions to provide reliable and effective transaction handling due to growing network demand [3].

Distributed ledger technology faces a looming challenge: BC bloat. As more transactions get etched onto the BC, it swells in size. This data explosion brings issues like storage woes, sluggish performance, and operational headaches. Imagine miners struggling to process a mountain of transactions, slowing down the entire network. That's why innovative bloat-taming strategies are crucial [4]. These solutions aim to find smarter ways to store and manage a growing volume of transactions, ensuring the BC remains efficient and effective even as it bulges with valuable data. The identification and investigation of strategies aimed at countering BC bloat is imperative in order to optimize resource utilization, enhance scalability, reduce expenses, ensure data integrity, enhance performance, and foster innovation within the BC domain [5]. Understanding these strategies and their usefulness in resource-constrained contexts, in particular, becomes critical in dealing with the shifting landscape of BC technology [3]. Accordingly, this study tackles this research gap and investigates these techniques through a review and critical analysis of the available literature and professional websites. This was accomplished by responding to the following research questions:

RQ1: What techniques have been introduced to combat the issue of BC bloat?
RQ2: What challenges are associated with techniques designed to alleviate BC bloat?
RQ3: Which of the BC bloat techniques suits resource-constrained devices?

This study makes several contributions.

- It enhances the existing body of knowledge regarding techniques for addressing BC bloat. The study delves into the various methods employed or proposed to reduce overall bloat, considering whether direct on-chain techniques are necessary or if off-chain alternatives can yield advantages. Two thematic solutions emerge from our analysis: on-chain techniques, which encompass BC structural modifications, pruning, sharding, ephemeral BC, zk-SNARKs, and off-chain techniques, including off-chain storage, light clients, and historical data storage.
- While these techniques prove effective in mitigating BC bloat, they also introduce certain concerns. This paper extensively examines these concerns, elucidating the strategies for researchers and practitioners to effectively navigate these techniques.
- Given that these techniques were originally designed to alleviate BC bloat, it becomes crucial to assess their applicability to resource-constrained devices. Such devices face even greater challenges in dealing with BC bloat issues. Several factors must be considered while determining the viability of BC bloat approaches for resource-constrained devices. Accordingly, some techniques like light clients, ephemeral, and sharding appear to be the most promising options for resource-constrained devices, compared to other techniques like zk-SNARK and off-chain storage.
- The findings of this study can catalyze future research endeavors, stimulating new investigations into BC optimization and bloat management. This paves the way for further innovation and progress within the industry.

The remainder of the paper is structured as follows: Section 2 provides background for the study, including causes for BC bloat and requirements. The method utilized to address the research questions is described in Section 3. Section 4 goes over the various BC bloat remedies. Section 5 covers the challenges of the suggested BC bloat remedies. Section 6 discusses the applicability of suggested techniques in resource-constrained devices. Section 7 summarizes the findings of this study and sheds insight into its limitations as well as future research directions. Finally, Section 8 brings this study to a conclusion.

2. Background

2.1. Blockchain bloat

The fundamental understanding of the BC bloat issue is inextricably linked to the underlying distributed nature of BC technology, which originates from Nakamoto's initial concept of facilitating financial transactions outside of bank or government control [6]. Several important elements were required for this vision: To begin, the network should continue to function even if a government or hostile actor shuts down one server, stressing decentralization. Furthermore, the network should have redundancy methods to guarantee that the failure of one server does not render the others inoperable [5]. In essence, each network node must keep a complete log of all BC transactions. Consequently, this implies that each node stores a complete record of all transactions within the BC, and as the BC grows with additional blocks, all nodes synchronize and store these blocks to maintain the distributed ledger. The rush of transactions leads to the generation of additional blocks as the BC network grows. To maintain the network's decentralized character, each full node must contain a complete set of these blocks [4]. Since BC meticulously records every transaction, it keeps getting bigger and filled to the brim with data; that's what we call BC bloat. This data pile-up creates a domino effect of problems: transactions crawl at a snail's pace, making them more expensive to process. Not only that, but joining the network becomes an ordeal, as new users have to download the entire history of the BC [3]. Bitcoin, for example, can take weeks just to get started. This data overload is a major roadblock for wider adoption of BC technology, as slow transactions turn potential users away.

BC bloat can result in a number of problems. To maintain a complete duplicate BC, more storage space is first needed. Second, additional bandwidth is needed to keep dispersed nodes in sync and to sustain communication [7]. This is particularly valid for recently added nodes to the network. Third, the time necessary to authenticate and verify all transactions in the chain rises as the BC expands, possibly influencing transaction processing delays. Fourth, fewer complete nodes will be available due to the high costs and requirements of data storage, which could result in centralization [8]. In essence, the centralization threatens the fundamentally decentralized nature of the BC. Finally, BC keeps everything, including errors and defects, which makes it more challenging to release a new version that fixes the original [9].

Table 1
Literature review findings.

Category	Study	Focus	Innovation
Blockchain scalability solutions	[8]	Enhancing BC scalability by combining IFS and BC data compression	Proposed a storage scheme
	[10]	Merging the IPFS with the dual BC	Proposed a new scheme to enhance BC scalability
	[11]	Combining BC and IFS to enable distributed BC storage	Proposed a distributed storage technique
	[12]	Enhancing BC performance	Proposed the ETS_GA technique
	[13]	Enhancing BC scalability	Proposed the Gridb technique
	[14]	Enhancing healthcare BC storage	Proposed the ParallelChain technique
	[15]	Enhancing data storage of industrial IoT based on sharding BC	Proposed a scalable data storage mechanism
	[16]	Enhancing BC scalability for IoT devices	Proposed a generic model
	[17]	Enhancing BC scalability	Proposed a parallel multiprocessor approach
	[18]	Improving BC scalability	Proposed using Setchain data-type
	[19]	Improving BC scalability	Proposed checkpoint mechanisms
	[20]	Enhancing BC data storage	Proposed the DSSBD approach
	[21]	Improving BC scalability	Proposed a method using a field-by-field basis BC transaction's construction
Blockchain storage optimization	[22]	Analyzing attacks on sharding storage	Proposed a tracing sybil attacks technique
	[23]	Enhancing data integrity in blockchain	Proposed a compactness methodology by integrating tiny blocks within standard BC configurations
	[24]	Solving blockchain scalability problem	Proposed a method using zK-SNARK
	[25]	Enhancing permissioned sharing and cross-organization data access	Proposed a P2P storage system on a consortium BC
	[26]	Enhancing security and privacy of BC	Proposed an approach using IPFS to store only hashes of signatures and public keys on the BC itself
	[27]	Deploying a proof of forest for sidechains using zk-SNARK technology	Proposed a technique for sidechains
	[28]	Asynchronous advantage actor-critic algorithm and AI to wrap data	Proposed a data management technique
	[29]	Investigated how to improve data sharing in BC sharding	Proposed an auditable data exchange
Review studies	[30]	Exploring BC scalability reasons and solutions	Storage affects public BC scalability
	[31]	Exploring BC data compression, pruning, and sharding	Data compression, pruning, and sharding techniques to enhance BC storage
	[32]	Exploring BC data compression, pruning, and sharding	Data compression, pruning, and sharding techniques to enhance BC storage
	[33]	Sharding approach to enhance BC scalability	Sharding technique to improve BC scalability
	[4]	Comparison between BC storage systems and cloud storage systems	BC to enhance traditional cloud storage
	[34]	Exploring cross-sharding, epoch randomization, shard reconfiguration, and node assignment	Cross-shard transaction processing, epoch randomization, shard reconfiguration, and node assignment techniques to enhance sharding
	[35]	Investigating the effectiveness of a lightweight indexing strategy	Lightweight indexing to enhance healthcare BC-based data storage
	[36]	Investigating how SegWit may enhance Bitcoin efficiency	SegWit technique can enhance BC scalability
	[37]	Assessing the costs of off-chain vs. on-chain BC-based storage	Comparative analysis of off-chain and on-chain costs
	[38]	Pruning techniques tailored for VANETs	Pruning to reduce bootstrap overhead
	[39]	Techniques to enhance BC scalability	Discussing both off-chain and on-chain approaches, assessing their latency, security, and cost implications
	[40]	Reviewing the challenges of BC performance	Sharding, lightning networks, layered solutions, and data compression techniques to enhance BC scalability
	[41]	Reviewing the BC scalability challenges	Sharding and off-chain techniques to enhance BC scalability
	[42]	Reviewing the BC storage optimization methods	replication-based, redaction-based, and content-based techniques to enhance storage optimization
	[3]	Reviewing the technique to improve storage in BC-based IIoT applications	Pruning, sharding, data compression, optimization techniques to enhance BC storage scalability
	This study	Reviewing current techniques for enhancing BC storage scalability	All techniques covered in previous reviews and new techniques, including historical data storage, ephemeral, and zk-SNARK techniques, to enhance BC storage scalability

2.2. Related literature

The literature can be categorized into three main themes: the presentation of novel solutions addressing BC bloat, BC storage optimization, and the review of existing literature aimed at identifying the causes and potential remedies for BC bloat, as outlined in Table 1.

- **BC scalability solutions:** Several studies have focused on providing BC scalability solutions. In [8], the authors investigated minimizing block size by combining IFS and BC data compression to improve BC scalability. Also, Mahmud et al. [10] looked toward improving BC scalability. By merging the IPFS with the dual BC idea, which boosts transaction throughput, the authors presented a new BC scheme [10]. A distributed storage technique based on BC and an Interplanetary File System (IFS) was presented by Zhang and Zhao [11] to encrypt voice data. The authors in Li et al. [12] introduced the Elite Tabu Search Genetic Technique (ETS_GA), a technique for secure server placement in edge computing, demonstrating superior performance in terms of convergence speed and identifying optimal solutions. Another technique called GriDB was presented by Hong et al. [13], which is a scalable BC database using sharding. Authors in Pawar and Sachdeva [14] proposed a ParallelChain healthcare architecture, dividing the network into clusters to linearly scale nodes and execute transactions in parallel, offering a more energy-efficient leader selection mechanism compared to proof-of-work. Also, Ren et al. [15] proposed a scalable data storage mechanism for industrial IoT based on coded sharding BC, which uses coded sharding technology to improve fault tolerance and storage load. It also designs a cryptographic accumulator-based data storage scheme to solve the security problem of data storage and verification [15].
- More recently, authors in Gurunathan et al. [16] analyzed BC platforms and proposes a generic model to enhance scalability for IoT devices. In [17], the authors suggested a parallel multiprocessor approach to solve BC scalability issues by increasing transaction throughput and reducing mining latency and cost, addressing performance bottlenecks. Setchain, a distributed data format enhancing scalability, was also proposed by Capretto et al. [18]. In [19], the authors also proposed a scalable BC consensus approach using checkpoint mechanisms. In [20], the authors introduced a Dynamic State Sharding-based BC Design (DSSBD), addressing cross-sharding transaction issues through graph segmentation and relay transactions, leveraging deep reinforcement learning for dynamic adjustment of shard parameters. Authers in Davies [21] introduced a method for constructing BC transactions using Merkle trees to verify transaction data on a field-by-field basis, deployable either at the system level or as a second-layer protocol without altering the underlying BC. This technique enables users to verify BC data efficiently by independently validating specific data items stored within transactions [21].
- **BC storage optimization:** Several studies have focused on BC storage optimization. The authors in Hafid et al. [22] researched sharding technology and its defense against Sybil attacks. The authors in Gracy et al. [23] introduced a compactness methodology to ensure data integrity in BC, proposing the integration of tiny blocks within standard BC configurations to minimize memory requirements, evaluated using Hyperledger Caliper. In [24], the authors presented a method leveraging Succinct Non-Interactive Argument of Knowledge (zk-SNARK) proofs to meet the requirements of trustworthy information storage systems, offering a faster, more reliable, and secure BC verification system. Authors in Peng et al. [25] proposed a P2P storage system on a consortium BC to enable trustworthy permissioned sharing and cross-organization data access, demonstrating its feasibility through implementation and testing. Moreover, authors in Thanalakshmi et al. [26] combined IPFS with post-quantum cryptography to expedite transaction processing by storing actual data on IPFS while keeping only hashes of signatures and public keys on the BC. This may increase the speed at which transactions are processed and decrease the required storage.
- More recently, in Bespalov et al. [27], the authors investigated how to create a proof of forest for sidechains using zk-SNARK technology. Zk-SNARK acts as a validation mechanism, confirming the BC's validity and removing the requirement for nodes to keep a large ledger up-to-date [27]. To achieve effective management of data in a BC-based IoT environment, the authors in Tsang et al. [28] applied the asynchronous advantage actor-critic algorithm to utilize AI with the best data wrapping and data reproduction strategies. According to the author, this method outperformed other common strategies and resulted in improving data management [28]. Moreover, by using BC sharding, the authors in Huang et al. [29] investigated how to improve data sharing. Using Zero-Knowledge Proof (ZKP) technology, the authors created an auditable data exchange method to safeguard vehicle identification privacy [29].
- **Review studies:** Several review papers were identified and included in this paper. For example, the authors in Khan et al. [30] conducted a review to investigate the public BC scalability issue. The authors identified that storage was among the significant factors that affect public BC scalability [30]. In [31], the authors conducted a literature review to answer the question, ``How can BC data storage scalability be enhanced?'' The authors described several techniques, including data compression, pruning, and sharding [31]. The authors in Liu et al. [32] reviewed the literature regarding BC storage strategies that ensure store sustainability. The methods covered in this paper include BC data compression, pruning, and sharding. Furthermore, as the authors in Hashim et al. [33] described, sharding can improve BC scalability. In [4], the authors reviewed BC storage systems compared to cloud storage systems. The author argued that BC storage can mitigate several traditional cloud storage issues due to its distribution nature [4]. The authors in Liu et al. [43] conducted a literature study on the sharding strategy for BC storage. Their investigation concentrated on determining the functional elements, including cross-shard transaction processing, epoch randomization, shard reconfiguration, and node assignment, among other components [34]. The authors in Singh et al. [35] investigated the effectiveness of a lightweight indexing strategy for healthcare data in BC. In [36], the authors investigated the problems with the Segregated Witness Implementation (SegWit) technique through a simulation experiment. The authors in Yadav and Gupta [37]

assessed the costs of off-chain vs. on-chain BC-based storage. The authors provided comparison scenarios showing that prices depend on the application.

- More recently, in Bowlin et al. [38], the authors discussed pruning techniques tailored for Vehicular Ad-hoc Networks (VANETs), aiming to preserve BC state while reducing bootstrap overhead. In [39], the authors reviewed the methods for enhancing BC storage scalability, including both off-chain and on-chain approaches, assessing their latency, security, and cost implications. The authors in Alshahrani et al. [40] reviewed the challenges faced by BC networks in processing a high volume of transactions. They explored potential solutions such as sharding systems, lightning networks, layered solutions, and compression-based methods. The authors in Rao et al. [41] reviewed the current BC scalability issues, including constraints like sharding and off-chain scaling. They explored layer 2 scaling and alternative approaches, highlighting their impact on BC applications. Key concerns include transaction throughput and network latency. Authors in Heo et al. [42] reviewed current BC storage optimization methods, categorizing them into replication-based, redaction-based, and content-based approaches. Replication-based strategies aim to minimize data duplication among participants, while redaction-based methods allow for modification or deletion of ledger data, and content-based approaches involve data compression.

The authors in Akrasi-Mensah et al. [3] investigated the issue of BC storage. The authors also identified the potential solutions for this problem, including data compression, summarization, and on-chain and off-chain storage [3]. According to these review papers' findings, the work in Akrasi-Mensah et al. [3] provides the most comprehensive review compared to other reviews; however, our review provided more comprehensive and up-to-date techniques used to mitigate BC data bloat issues, which were not covered in the previous reviews. So, our paper can be considered an extension of the work in Akrasi-Mensah et al. [3], but we discuss more innovative techniques such as historical data storage, ephemeral, and zk-SNARK.

3. Research method

A multimodal method was used in this study to acquire and assess relevant literature addressing the solutions and challenges of these solutions linked to BC bloat. The first step entailed extensive research in respected scientific databases such as Science Direct, Wiley, Springer, Emerald, IEEE, and MDPI for all literature written in English. Even though these platforms are often significant sources of academic research, it became clear that the current corpus of work on BC bloat was very scant. As a result, supplemental

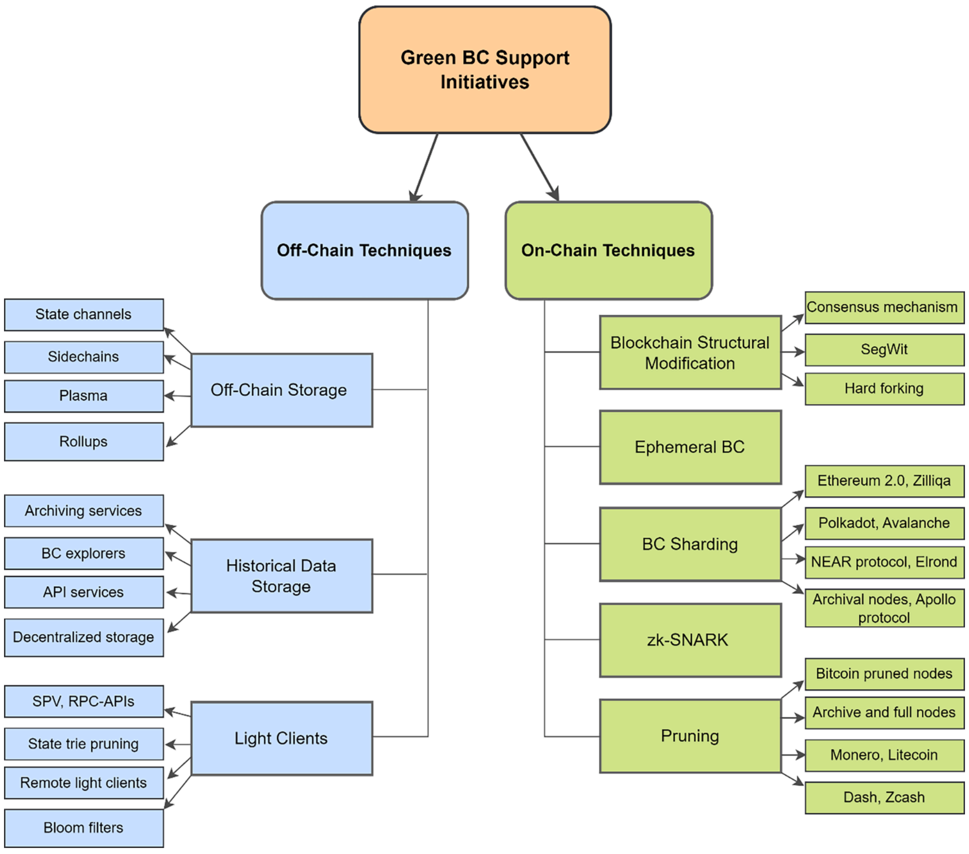

Fig. 1. Techniques to address BC bloat.

sources other than scholarly databases were used to get a more comprehensive grasp of the issue. Accordingly, we scoured professional websites, blogs, industry publications, and even dusty content repositories, gathering real-world experiences, opinions, and the latest buzz on this data dilemma. These sources, often overlooked by academics, proved to be a treasure trove of insights. We found practical perspectives, use cases, and even ingenious solutions that hadn't yet made their way into scholarly circles. To avoid potential bias in resource selection, we threw a wide net of keywords at our search engines, such as ``blockchain AND bloat AND solutions OR remedies OR scalability OR challenges OR obstacles OR drawbacks''. Each researcher did their own independent search. This double-checking helped us cast a wider net and gather the most objective data possible.

4. Techniques to solve blockchain bloat issue

This segment focuses on answering RQ1, which pertains to the techniques designed to mitigate BC bloat. Several strategies have been suggested and put into action to tackle BC bloat, encompassing pruning, historical data storage, sharding, ephemeral, zk-SNARK, off-chain, and light client solutions. In Fig. 1, a range of strategies are depicted, accompanied by instances of their implementation. These methods can be broadly classified into two categories: layer 2 BC, or off-chain methods, and layer 1 BC, or on-chain methods [9, 39].

4.1. On-chain techniques

On-chain solutions include techniques like consensus techniques, sharding, and adjusting block sizes. Most of the on-chain techniques used to reduce the problem of BC bloat are covered in this section.

4.1.1. Blockchain structural modification

The main goals of structural modification techniques are to enhance the fundamental characteristics and features of the BC network, like lowering the block verification time or raising the block size limit. To explain the implementation concepts here, for example, the Segregated Witness (SegWit) technique has to meet some requirements [30]. 1) Every bit of transaction data in a Bitcoin transaction is divided into two independent parts: the transaction data itself and the witness data. Details about the account balance are contained in the transaction data, and the witness data is used to verify the user's identity. For example, when receiving a transfer, the recipient just needs to determine whether the asset is available; they do not need to know any information about the donor. But under the Bitcoin transaction structure, the witness data, which includes signature data, takes up a lot of storage space, which hinders transfer efficiency and increases packing costs. 2) SegWit separates the witness data from the transaction data and stores it in a separate location, hence speeding up transaction processing [36]. 3) Witness data is not included in the transaction hash computation because SegWit deploys a unique Merkle tree structure for transaction data. By eliminating signature data from the transaction hash computation process, SegWit brings about a substantial modification to the signature hashing method. This modification makes better use of block space and optimizes transaction size [44]. Hard forking, SegWit, and consensus mechanism modification are examples of common structural modification techniques.

- Consensus mechanism modification: A BC consensus mechanism is the technique to validate transactions, which enables the network's security and accuracy [45]. Bitcoin, for example, has a Proof-of-Work (PoW) algorithm that necessitates massive computing power to resolve a difficult calculation required to record the block in the BC. Structural modification techniques contribute to improving underlying protocols (for example, Bitcoin's PoW) by altering how they handle data. Several recommendations suggest employing consensus mechanisms with fewer data requirements, such as Delegated PoS (DPoS) and Proof-of-Authority (PoA).
- SegWit focuses on altering the format and methodology of data storage inside the Bitcoin BC network [9]. It helps remove the signature data associated with every transaction, which increases transaction capacity and storage space. It is important to remember that around 70 % of the total space in a transaction is occupied by the digital signature used to confirm the sender's ownership and availability of funds [46].
- A technique called ``hard forking'' aims to fundamentally or structurally alter the characteristics of a BC network. A hard fork may include decreasing the time necessary to create a block or raising the block size. While hard forking is required for structural modification techniques, the most productive option is a controversial hard fork. The contentious hard fork symbolizes a rift within the greater BC community, with certain members of the community disagreeing on important issues [9].

4.1.2. Blockchain sharding

The process of shattering the BC into more manageable, smaller chunks is known as sharding [30]. A portion of the BC's data is contained in each shard, which lessens the strain on individual nodes. There are several implementation concepts for the sharding technique [31–33]. 1) The sharding technique divides the network into more manageable, smaller groups called shards, each of which holds a fraction of the blockchain's information and transaction history. Every shard functions independently, using its own consensus method to verify and authenticate transactions inside its domain [30]. 2) It requires effective cross-shard communication protocols that allow transactions across different shards to be seamlessly coordinated. 3) It requires controlling the global state of the BC, which includes account balances and smart contracts, which is another essential concept. 4) It also requires a dynamic shard allocation process, which allows the network to build or merge shards dynamically in response to demand and resource availability, maximizing workload distribution and network efficiency [44].

In BC technology, sharding is a scaling technique that boosts network performance and transaction speed. The congestion and delay caused by processing every transaction on a single chain can be reduced by allowing each shard to handle its own set of transactions and smart contracts independently [33]. Large-scale applications and platforms that seek to rival established financial and data processing systems should pay special attention to this. Here are some actual instances of BCs using sharding in the real world [22,34].

- Ethereum 2.0: Sharding technique is used by Ethereum 2.0 to increase network scalability. Ethereum can process many more transactions per second thanks to the handling of individual contracts and transactions by each shared chain.
- Zilliqa: The BC platform, Zilliqa, was designed from the ground up with sharding in mind. Integrating a sharding technique with the PBFT consensus process increases network speed.
- Polkadot: A multi-chain network called Polkadot connects numerous BCs to form a single ecosystem. It manages several BCs termed ``parachains'' using a sharding method.
- Avalanche: Avalanche is a consensus platform with a novel approach to sharding. Instead of employing prefabricated shards, it creates subnetworks, known as ``subnets,'' dynamically based on network requirements.
- NEAR protocol: NEAR protocol divides its network into multiple shards, each capable of processing its transactions and smart contracts.
- Elrond: Elrond divides the network into multiple shards, each with its consensus group and validator set.

4.1.3. *Ephemeral blockchain*

Ephemeral BCs can provide a solution to the bloat problem. Several implementation concepts underpin the Ephemeral BC approach. 1) It concentrates on temporary or short-term data storage, where BC records are kept momentarily rather than permanently [44]. 2) It chooses data by applying preset standards such as transaction importance or relevancy. Ephemeral BC places a higher priority on the retention of data that immediately benefits the blockchain ecosystem than on archiving all transaction data. 3) It frequently makes use of effective data-pruning methods to eliminate unnecessary or out-of-date information from the BC ledger. 4) It could remove unnecessary data from the core BC network by utilizing off-chain storage options.

An ephemeral BC is one in which the blocks do not linger forever on the chain but are discarded as soon as the function of the BC recording the transactions is completed [7]. This method reduces the quantity of data that each node must load [45]. Ephemeral BC is temporary in nature, initiating with the creation of the genesis block, which serves as the first building block in the chain. When the BC reaches a predetermined data capacity, a specific cut-off point is established to mark the culmination of its primary purpose [5].

The fusion of the ephemeral approach with BC technology presents the opportunity to establish a secure and private digital ecosystem that combines the strengths of both domains. In this context, sensitive data finds its home on the BC, guaranteeing its security and immutability. At the same time, the ephemeral approach imposes time-limited access to this data, thereby diminishing the risk of unauthorized access and potential data breaches [45]. The procedure may involve retiring the original BC and, if necessary, reconstructing it using the archived data from the database. Storage and retrieval of data from transient BCs can be facilitated using on-premise or cloud solutions. This approach ensures the secure storage of information within the database while conserving valuable space on the main BC by eliminating redundant and outdated information [7].

4.1.4. *Utilizing zk-SNARK*

Another technique that can provide an on-chain solution to the bloat problem is the zk-SNARK technique. Several implementation concepts underpin the zk-SNARK approach [24,27,29]. 1) zk-SNARK makes use of zero-knowledge proofs, which enable the prover (a party) to convince the verifier (a different party) that a statement is true without disclosing any further information beyond the veracity of the assertion. 2) Because it is intended to produce concise proofs, even for intricate computations, the proofs themselves are often brief and easily verifiable. 3) It is non-interactive, which means that back-and-forth contact between the prover and verifier is not necessary for the evidence production process. 4) Its foundation is the idea of knowledge arguments, in which the proponent proves that they are aware of a certain piece of information while withholding the information itself.

The zero-knowledge zk-SNARK is built upon the concept of Zero Knowledge Proof (ZKP), which allows entities to verify possession of specific data without divulging that data [5]. This concept is similar to a user or a node needing to prove ownership of a password to access a website without actually revealing the password [27]. An example of a BC that incorporates zk-SNARKs is the Coda BC. Coda BC condenses its total state into a compacted 1 KB zk-SNARK proof. This proof serves as a validation mechanism, affirming the integrity of the BC and eliminating the need for nodes to maintain an extensive ledger [27].

4.1.5. *Pruning*

The pruning technique aims to diminish the storage requirements of BC nodes without compromising transaction validation. Several implementation concepts underpin the pruning technique [31,32,47]. 1) Pruning keeps essential data in the BC while removing superfluous data on a selective basis. 2) It retains crucial historical data required for transaction validation and network consensus maintenance, even if it eliminates certain data from the BC. 3) BC data compression methods are frequently used in pruning. 4) Pruning architecture is consistent with the BC network's consensus method. By doing this, it is made sure that nodes that have been pruned may continue to successfully take part in consensus processes even after some data has been deleted from their local copies of the BC.

By trimming less relevant data, BC nodes can significantly reduce their storage needs. This becomes especially crucial as pruning allows a broader spectrum of users to operate nodes without the necessity for extensive storage capacity, making it an integral component of BC networks [38]. Importantly, the act of pruning nodes does not diminish their capability to enhance network security;

they retain the ability to verify blocks and transactions. Meanwhile, full nodes, holding the complete BC history, continue to provide access to the data that was removed during the pruning process. These are a few BC applications that benefit from pruning [5,48].

- Bitcoin pruned nodes: On pruned nodes, only a partial subset of the BC data is retained. These nodes specifically hold the most recent transaction data to facilitate the validation of new transactions and blocks. Although older transaction data remains accessible from other nodes within the network, it is not stored locally on pruned nodes. Consequently, maintaining a Bitcoin node requires significantly less storage, making pruned nodes essential for individuals with restricted bandwidth or limited storage capacity.
- Ethereum archive and full nodes: The Ethereum network accommodates numerous nodes with diverse storage requirements. The archive nodes maintain the selected data, while the full nodes maintain the full data record. This enables balance among full nodes between interaction and storage, which enhances the scalability of the Ethereum network.
- Monero pruning: In this method, only essential data required for both transaction validation and privacy protection is maintained on the node while other old data is removed.
- Litecoin: Litecoin counts on "pruning nodes" to selectively retain a small portion of BC data. This ensures the efficient functioning of the network, particularly for resource-constrained nodes like IoT devices, while enhancing decentralization capabilities.
- Dash pruning: The Dash cryptocurrency is described for its rapid transaction processing, in which pruned nodes can validate transactions swiftly without the necessity of retaining the entire transaction history by maintaining only selected data from the entire BC.
- Zcash: Zcash employs a node pruning method to enhance efficiency, which empowers nodes to diminish their storage requirements.

4.2. Off-chain techniques

Off-chain techniques can be broadly categorized into three main themes: off-chain storage transaction processing or state channels, parallel BC or side chains, and lite clients. This section discusses these techniques, including some examples of each technique.

4.2.1. Off-chain storage

One of the off-chain techniques is off-chain storage. Several implementation concepts underpin the off-chain techniques [3,11]. 1) Choosing which non-essential data to move off the primary BC is the first step in off-chain storage. 2) Off-chain storage options use dependable storage techniques to guarantee the accuracy of data sent away from the primary BC. The safe storage of off-chain data may entail the use of distributed storage systems, encrypted databases, or reliable third-party storage suppliers. 3) To guarantee consistency between data on-chain and off-chain, off-chain storage solutions uphold synchronization protocols. 4) Effective data retrieval techniques are given top priority in off-chain data storage systems to provide prompt access to stored data when required. For quick and dependable access to off-chain data, this may entail putting in place indexing, caching, or retrieval methods. 5) Independent processes are frequently used in off-chain storage solutions to settle transactions. By offloading some of the transaction processing and smart contract execution from the main BC layer, off-chain storage aims to increase the scalability of BC networks [48]. Here are some real-world instances of off-chain storage and processing techniques [3,4,48].

- State channels: State channels enable off-chain transactions involving two or more participants. Only the completed state of each off-chain transaction is recorded on the BC, allowing participants to engage in multiple off-chain transactions. For use cases requiring low latency and minimal fees, such as micro-transactions, these channels are perfect. For instance, Lightning Network is an off-chain Bitcoin technique that enables users to create payment channels and carry out speedy, affordable off-chain Bitcoin transactions [10]. Channels can be started, multiple transactions can be made within them, and only the net result is recorded on the Bitcoin BC. Certain BC platforms come with sidechain functionality integrated right in. For example, the Loom network offers a framework for creating Ethereum sidechains. Developers can create sidechains for games, collectibles, and decentralized apps (DApps) to offload transaction processing from the Ethereum mainnet.
- Sidechains: Sidechains are distinct BCs that can communicate with the main BC but have different rules, consensus procedures, and features. Sidechains provide higher transaction throughput by allowing transactions to be executed in parallel on both the main BC and the sidechain. Rootstock (RSK), for example, is a smart contract platform that functions as a sidechain to Bitcoin BC.
- Plasma: Plasma is a framework that permits the formation of ``plasma chains,'' or child chains, that are linked to the main BC. These chains execute transactions independently and transmit aggregated data to the main BC regularly. Plasma can considerably boost the main chain's transaction throughput. The OmiseGO (OMG) network, for example, uses a plasma-based approach to improve Ethereum's scalability. The OMG network prioritizes quicker and cheaper value transfers as well as decentralized exchange services, while depending on Ethereum's security for finality.
- Rollups: Before sending transactions to the on-chain BC, rollups bundle and optimize them. Rollups are classified into two types: optimistic rollups and zk-rollups. To assure data authenticity, optimistic rollups rely on fraud proofs, whereas zk-rollups utilize zero-knowledge proofs for rapid verification. Optimistic Ethereum (Optimism), for example, is an optimistic rollup technique for Ethereum. It attempts to boost Ethereum's scalability by processing transactions off-chain and employing a fraud-proof method to assure transaction authenticity.

4.2.2. Historical data storage

Another technique for off-chain solutions is historical data storage. Several implementation concepts underpin the historical data storage techniques [3]. 1) To decrease the size of the main BC, this technique includes separating data that is older or less regularly accessed [36]. 2) The historical data is kept apart in systems designed specifically for archival purposes, which makes it easy to preserve and retrieve vast amounts of historical BC data. 3) Indexing methods are used to arrange archived BC data according to pertinent features such as transaction ID or timestamps to efficiently retrieve and query past data. 4) To limit storage space needs and lower storage costs, this technique frequently incorporates optimization of data and compression methods. Data may be condensed using compression methods without sacrificing its integrity. 5) The technique may also deploy data management strategies to control the preservation, archiving, and disposal of historical BC data in accordance with predetermined standards such as data relevancy, legal obligations, or storage capacity limitations. By eliminating unnecessary or duplicate data, these regulations guarantee that only pertinent historical data is kept on file, freeing up storage space.

Full nodes can concentrate on recent or critical data, but older data can be retrieved as needed [35]. Historical data storage techniques are critical for a variety of use cases, including smart contract audits, assessing BC operations, and developing apps that rely on previous BC data. They enable the accessibility and availability of historical records, which is especially important for BC networks that prune older data to preserve efficiency and reduce BC bloat. These solutions make BC technology more usable and serve a wide range of applications and services that rely on historical data. The following are instances of real-world uses of historical data storage techniques [3,4,35,45].

- Full node archives: Full node archives include an exhaustive record of all historical data on a BC, including every transaction, contract status, and block, while full nodes may delete old data to enhance performance.
- BC Explorers: BC Explorers, which are used as tools on the web, organize and display historical BC data in a user-friendly manner such that users can explore the full data by searching for certain transactions. Etherscan, deployed by Ethereum, is an example of a BC explorer that enables searching transactional BD historical data.
- API services: API services enable developers to retrieve historical BC data. For example, Alchemy, a BC developer platform, offers API access to Ethereum data.
- Decentralized storage networks: Several BC initiatives strive to decentralize the storage of historical data. As an illustration, immutable BC files can be distributed and stored on the decentralized Filecoin network, ensuring the enduring accessibility of historical BC data.
- Archiving services: Such services contribute to maintaining a dependable historical record, particularly for BCs that adopt practices like removing outdated data. For instance, Amberdata is a BC data provider that offers archiving services to keep complete historical data for different BCs like Bitcoin, Ethereum, and others. Researchers and developers use these services to obtain historical BC data.

4.2.3. Light clients

The third technique of the off-chain solution is the light client. Several implementation concepts underpin the light client techniques [49]. 1) Light clients are made to save as little data locally as possible by simply keeping the ID and header, for example, necessary for accessing the BC network and authenticating transactions. 2) This technique uses streamlined procedures to confirm a transaction without requiring the completion of computationally demanding operations like consensus involvement or complete transaction validation. To guarantee transaction truthfulness, this technique relies on condensed validation techniques and cryptographic proofs. 3) To obtain pertinent BC data, light clients are linked to distant full nodes or reliable services rather than locally maintaining the complete BC ledger. They communicate with these complete nodes to get transaction details, confirm block headers, and keep their local state up-to-date with network changes. 4) Light clients frequently use the streamlined payment authentication procedure to confirm that transactions are included in blocks without downloading the complete block. This can be done, for example, by downloading only block headers and Merkle proofs. 5) Light clients use data compression and lightweight approaches to reduce latency and consume less bandwidth, which optimizes network connections. 6) Light clients are made especially to run well on devices with limited resources, such as IoT devices. 7) To guarantee accurate and dependable BC synchronization, light clients use strong synchronization methods, such as checkpointing and reorganization detection.

Wallet apps or nodes that don't save all of the BC are known as light clients. Rather, they access BC data through reduced verification techniques, which makes them appropriate for low-resource devices. Light clients, sometimes referred to as lightweight clients, are a class of solution that makes it possible to interact with BCs more effectively, especially for apps and devices with constrained storage and processing power. The examples below show how this technique can be used in the real world [35,49].

- Simplified Payment Verification (SPV): A common type of light client used primarily in Bitcoin and Bitcoin-based BCs is called SPV. SPV clients just request block headers, which include proof of the inclusion of a single transaction, rather than downloading the entire BC. This approach reduces bandwidth and storage requirements significantly. For example, mobile wallets like Electrum use SPV to let users manage their Bitcoin funds without having to download the entire BC. To verify transactions and verify the balance, they employ SPV.
- State trie pruning: With this approach, vital contract and account information is kept in the BC's state trie, but older, less frequently accessed data is removed. For instance, Ethereum light clients employ state trie pruning to provide instant access to BC state data, enabling apps to interact with Ethereum networks without keeping track of the entire state history.
- Remote Procedure Call (RPC) APIs: RPC APIs are available for many BCs, allowing developers to access BC functions and data on distant servers. These APIs allow applications to transfer processing tasks and storage needs to remote nodes. RPC APIs are used, for

instance, in Web3.js with Ethereum communication with Ethereum nodes. Web3.js enables DApps to interact with the Ethereum BC without requiring the hosting of an entire node.
- Bloom filters: Data structures called bloom filters allow light clients to ask a remote node whether certain data elements are available in the BC without revealing specific data details. For instance, Bitcoin Lite clients use Bitcoin Bloom filters to ask a distant node if there have been any transactions involving particular addresses without disclosing private information.
- Remote light clients: Some services provide remote light clients, which outsource resource-intensive operations to a distant server, while lightweight clients interface with this server to obtain BC data and conduct actions. As an example, Infura, in collaboration with Ethereum, provides a remote light client service. Infura, which is used by developers to link their Ethereum apps to remote Ethereum nodes, can decrease computational and storage requirements on their end.

Light clients are critical in making BC technology more accessible and practical for a wide range of applications, letting them operate effectively even on devices with limited resources. These techniques help to increase the use of BC technology across sectors and use cases. Some of the applications that may benefit from light clients are discussed as follows [35,49]:

- Mobile wallets: Trust Wallet (for Ethereum and Binance Smart Chain) and MyEtherWallet are two real examples of mobile cryptocurrency wallets. They employ light clients to let consumers manage their Bitcoin assets from their mobile devices.
- Web-based DApps: Light clients are frequently used in decentralized systems, such as those developed on Ethereum, to allow users to interact with smart contracts and BC data directly from their web browsers without running a complete node.
- BC Explorers: Light clients are used by services like Etherscan to provide users with an easy and convenient method to view BC data and transactions without the requirement for a full BC download.
- BC APIs: Light client techniques are frequently used by developers creating BC apps to interface with BC networks to simplify transactions, data retrieval, and interaction with smart contracts.
- IoT devices: Light clients can be used by IoT devices with minimal processing capabilities to connect with BC networks for use cases such as supply chain tracking and device identity management.

5. Challenges and recommendations of BC bloat techniques

This section delves into the particular challenges related to each of the potential BC bloat solutions. We also discuss potential future research directions and make recommendations for effectively addressing these challenges.

5.1. Challenges of BC bloat techniques

This section answers the RQ2 (the challenges of the techniques developed to address the BC bloat issue). BC bloat techniques encounter various challenges, which can be classified into 18 overarching themes. These themes encompass aspects like security and privacy, considerations of trust, issues related to data loss, effective data management, ensuring data availability, managing complexity, resource utilization, the choice of consensus mechanisms, recognizing case-specific limitations, scalability concerns, potential centralization, interoperability challenges, latency issues, economic considerations, storage costs, synchronization challenges, shared transactions, and functionality concerns. However, depending on the technique, the individual challenges may differ in degree or form. For each technique, a summary is provided in Table 2.

Table 2
Challenges of BC bloat techniques.

Challenge	Technique							
	Structural modification	Sharding	Ephemeral	Zk-SNARK	Pruning	Off-chain	Historical data	Light clients
Security & privacy	✓	✓	✓		✓	✓	✓	✓
Trust considerations				✓		✓		
Data loss		✓	✓		✓			
Data management	✓						✓	
Data availability					✓			✓
Complexity		✓		✓	✓	✓		
Resource consumption	✓			✓				✓
Consensus mechanism			✓					
Case limitations			✓					
Scalability				✓		✓	✓	
Centralization								✓
Interoperability						✓		
Latency	✓					✓		
Economic considerations						✓		
Storage cost							✓	
Synchronization			✓					✓
Shared transactions		✓						
Functionality								✓

5.1.1. Blockchain structural modification

BC contains a record of every transaction in sequential order. A fork of the BC could be necessary to update it to scale, which could lead to conflict among the BC's supporters. The scaling update can occur by forking the code; however, doing so results in two networks operating concurrently (such as Bitcoin and Bitcoin Cash). This may perplex people and lower the value of cryptocurrencies as a whole. The transition to a new consensus method may pose security problems. Developers may need to put in a substantial amount of time and effort to accomplish the modification. SegWit adoption, on the other hand, may be sluggish since it requires wallet and network support. Backward compatibility for those who do not upgrade might be difficult to manage [36].

5.1.2. Sharding

One major worry about Sharding is data loss. Sharding entails breaking the BC into smaller shards, with nodes within a shard handling just transactions for that shard [34]. This technique may result in data loss for nodes that do not preserve the complete BC, making past data difficult to obtain. This limitation could present a substantial drawback in scenarios where retaining a comprehensive transaction history is crucial. Another noteworthy challenge emerges with the impact of sharding on security. Since nodes within a shard possess only a partial view of the network, security concerns, such as the ``nothing-at-stake'' dilemma, become apparent, which results from validating multiple chains without penalty, creating potential vulnerabilities.

Another challenge results from the consensus process and software development in a sharded BC system [22]. Here, complex algorithms and protocols are required to enable efficient communication between shards. Moreover, multiple shard transactions (i.e., cross-shard transactions) pose an additional challenge, as these transactions require extra coordination, potentially causing delays that can adversely affect network efficiency and performance. In the case of resource-constrained devices, sharding might elevate network latency. In this case, these devices must communicate with multiple shards, potentially hindering transaction processing and diminishing overall network performance [22].

5.1.3. Ephemeral

Ephemeral BCs pose specific challenges, including synchronization issues, security risks, and potential data loss [7]. The ephemeral approach may present a significant threat to applications reliant on historical data, audits, or regulatory compliance. Security and provenance also represent additional notable concerns [45]. Deleting BC data can give rise to issues related to data security and provenance, especially in applications where an immutable and tamper-proof transaction record is imperative. Furthermore, frequent resets in ephemeral BCs may lead to synchronization problems [45]. Additionally, when new devices or nodes join the network, catching up to the BC's current state might be challenging, particularly for resource-constrained devices. Implementing ephemeral BCs also necessitates the establishment of specific consensus procedures to accommodate these recurrent resets. It's essential to note that ephemeral BCs are most suitable for scenarios prioritizing instantaneous validation and real-time processing over transaction history. Consequently, they may not be ideal for situations where a complete and unaltered transaction history is a requisite [45].

5.1.4. zk-SNARK

zk-SNARKs, being highly intricate mathematical structures, present challenges in both their development and evaluation, demanding specialized expertise [27]. The complexity involved, especially in resource-limited scenarios where simplicity and ease of implementation are prioritized, may hinder widespread adoption. Additionally, the utilization of zk-SNARKs requires complex cryptographic computations, such as polynomial evaluations and pairings of elliptic curves. Given that many IoT and fog devices operate with limited processing power and energy resources, these computations can pose a considerable burden on such devices [50]. While zk-SNARK proofs are notably shorter than standard proofs, they can still be sizeable, especially for intricate claims. In contexts with constrained resources, where storage and communication capabilities are often limited, this poses a significant challenge. Another critical issue is the dependable configuration needed for the initial generation of zk-SNARK parameters [24]. If the trusted setup is not managed correctly, it can give rise to trust issues and compromise the integrity of the entire system. Scalability is also a concern with zk-SNARKs [48], particularly in resource-constrained areas like fog networks and the IoT, where the computational and communication overhead associated with zk-SNARKs may impede smooth scalability.

5.1.5. Pruning

Deleting data permanently results in the potential loss of historical data. This could be a drawback when a complete transaction history is essential for legal, auditing, or compliance purposes. Moreover, pruning may impact the security environment [38]. Nodes that eliminate historical data face the risk of losing track of attempted fraud or double-spending, and in certain situations, this historical data may be crucial for identifying and resolving security vulnerabilities. Pruned BC networks also encounter challenges related to data availability. The decentralization aspects of BC networks, which rely on full nodes to furnish historical data, may be affected by pruned nodes' inability to serve later-joined nodes or users [47]. Finally, the technical implementation of pruning may be complex, especially in BC networks with intricate data structures. Ensuring that the pruning procedure is error-free and does not compromise the integrity of the BC may present a challenge [3].

5.1.6. Off-chain storage

Centralization poses a notable challenge in off-chain storage methods, often relying on intermediaries or off-chain service providers for transaction completion. This introduces counterparty risk, jeopardizing the trustless structure inherent in BC technology [37]. Security becomes an additional concern, particularly for centralized off-chain systems susceptible to fraud, security lapses, and suboptimal intermediary management. Interoperability represents another significant challenge, as ensuring compatibility between various off-chain methods and diverse BC platforms can be difficult, limiting their widespread adoption. The complexity of the implementation and management of sidechains and state channels may render them less accessible for developers and users [37]. Despite efforts to reduce latency and enhance transaction throughput, latency remains a concern, posing potential obstacles in scenarios where real-time response is crucial, such as in the IoT and fog computing. Furthermore, the economic model of off-chain techniques, which includes fee structures and intermediary incentives, may not always align with the goals of IoT and fog applications. This misalignment can present challenges in integrating off-chain techniques into these specific contexts.

5.1.7. *Historical data*

Historical data storage methods face a primary and critical challenge related to data volume [35]. The preservation of massive amounts of data through these methods poses significant storage and management challenges, particularly for nodes and devices with limited resources [4]. Another major hurdle is the cost of storage [37], especially for resource-constrained devices that find it expensive to store substantial historical data on-chain. Scalability issues also arise, particularly in resource-constrained environments like IoT networks, as they expand. The management of vast amounts of historical data on a BC can be challenging, impacting overall network performance [11]. Security and privacy represent major concerns in storing historical data on a public BC [35]. Compliance with privacy standards such as GDPR may restrict long-term data preservation, further complicating matters. Lastly, the retrieval of data from a BC may involve delays. This drawback could impede real-time access to historical data, often essential for fog and IoT applications [3].

5.1.8. *Light clients*

The primary challenge with light clients lies in their inability to autonomously validate transactions, smart contracts, or blocks [49]. They rely on complete nodes for validation, introducing several difficulties, with security being a major concern. Hostile full nodes could exploit vulnerabilities in light clients, compromising their security and privacy [35]. If a majority of full nodes are compromised, the security and privacy of light clients could be significantly jeopardized, especially considering the potential for full nodes to log light client activity, putting user privacy at risk. Dependence on a limited number of full nodes raises concerns about the possibility of centralization. Although light clients have lower resource requirements than full nodes, they still require memory, processing power, and network connectivity, which may pose challenges, particularly when efficient resource management is crucial [35]. Synchronization delays are an additional consideration for light clients, as they rely on full nodes for BC updates [49]. This can lead to delays in real-time applications and services, especially when discussing implications for the IoT and fog computing.

5.2. *Recommendations to address the challenges of BC bloat techniques*

To successfully tackle BC bloat, it is crucial to address the associated challenges. This section explores various approaches, highlighting specific methods where applicable. The summarized findings are presented in Table 3, and further details are discussed in the following paragraphs.

Addressing security and privacy concerns in the context of BC bloat involves strategic choices such as adopting off-chain solutions and zk-SNARKs. While zk-SNARKs provide a robust cryptographic foundation for privacy, they employ complicated cryptographic

Table 3
Recommendations and potential techniques used.

Challenge	Recommendation	Potential application	Study
Security & privacy	• Implement advanced cryptographic protocols to enhance security	zk-SNARKs, off-chain, structural modification	[27,36]
Trust considerations	• Deploy trustless principles BC bloat technique	zk-SNARK	[27,37]
Data loss	• Implement a tiered storage system • Ensure a balance between data retention and BC size	Pruning, historical data storage, ephemeral	[45,47]
Data management	• Develop data management policies	Pruning, historical data storage, off-chain	[38,47]
Data availability	• Maintain a network of archival nodes	Pruning, sharding, ephemeral	[38,45]
Complexity	• Enhance user-friendly tooling	zk-SNARK	[24,48]
Resource consumption	• Opt for a lightweight consensus mechanism	Pruning, zk-SNARK, structural modification	[27,48]
Consensus mechanism	• Tailor consensus mechanisms to the needs of the specific BC bloat technique	All techniques	[5,46]
Case limitations	• Evaluate the suitability of specific BC bloat techniques	All techniques	[5]
Scalability	• Leverage sharding	Sharding	[22,34,41]
Centralization	• Minimize centralization risks in off-chain solutions	off-chain	[37]
Interoperability	• Develop standardized protocols	Off-chain, structural modification	[37]
Latency	• Implement efficient cross-shard communication protocols	Sharding	[22,34]
Economic considerations	• Define clear economic models	Off-chain	[37]
Storage cost	• Explore off-chain storage options	Pruning, historical data storage, off-chain	[37,45]
Synchronization	• Develop lightweight synchronization protocols	Light client, off-chain	[49]
Shared transactions	• Ensure robust security mechanisms and consensus protocols	Off-chain	[22,34]
Functionality	• Define the functionality and scope of light clients	Light client	[49]

protocols to enhance security [27]. On the one hand, sophisticated security and consensus mechanisms are required for shared transactions [37]. On the other hand, off-chain solutions require robust privacy measures to safeguard sensitive data. Additionally, exploring on-chain historical data storage options may contribute to long-term reliability [37]. Moreover, BC bloat techniques must be compatible with trustless design principles to enable trust. Techniques like zk-SNARKs maintain transactions' trustlessness by enabling verification without revealing sensitive information [24].

Mitigating data loss challenges involves implementing a tiered storage system that incorporates strategies like pruning and historical data storage. Moreover, by establishing clear data management policies, a balance between data retention and BC size can be achieved [45]. Furthermore, archive nodes can enhance data availability, which is particularly significant for methods like sharding and pruning, as archival nodes contribute to the decentralization of the BC network [41]. Additionally, preventing undue centralization in the network requires minimizing risks in off-chain solutions through decentralized validation and channel management [38].

The complexity of certain algorithms, such as zk-SNARKs, can be mitigated by providing comprehensive documentation and user-friendly tooling. This approach aims to reduce adoption barriers, making these tactics more accessible to a broader audience [27]. Moreover, mitigating resource consumption involves selecting consensus mechanisms suitable for devices with limited resources. Efficient consensus mechanisms benefit pruning, zk-SNARKs, and light client techniques. They also offer an advantage for resource-constrained devices [27]. Introducing new consensus procedures gradually ensures user adaptation and provides fallback options in case of issues.

Tailoring consensus mechanisms to the requirements of specific BC bloat techniques contributes to resolving complexity challenges. Effective and resource-constrained, context-optimized consensus algorithms enhance the overall efficacy of the chosen technique. Scalability is achieved through sharding, splitting the BC into smaller shards while maintaining data integrity. Effective cross-shard communication strategies are essential to preserving data integrity and consistency amongst shards, particularly in IoT environments [34]. Improving interoperability involves standardizing interfaces and protocols for different off-chain solutions. This facilitates smooth communication between various BC platforms and off-chain components, especially beneficial in heterogeneous IoT and fog computing environments [37]. Mitigating latency issues in sharding solutions requires implementing efficient cross-shard communication protocols. Prioritizing low-latency networking technology in IoT ensures real-time data delivery [22]. Also, addressing synchronization challenges involves developing lightweight synchronization methods for light clients, eliminating latency for real-time access to previous BC data [35]. Furthermore, precision in defining the scope and functionality of light clients, considering their constraints, is critical. Establishing clear boundaries for light client functionality is essential, particularly in cases where limited validation capabilities are acceptable [22].

Addressing case-specific limitations involves evaluating suitable BC bloat approaches based on the project's goals and specifications. Assessing each technique's advantages and disadvantages concerning the use case leads to optimal results [5]. Additionally, clear economic models are essential for off-chain solutions, encompassing fee structures and incentives for intermediaries. Ensuring consistency with application aims and financial capabilities is critical for the success of off-chain solutions [37]. Off-chain storage possibilities may help decrease on-chain storage costs, which is particularly crucial for resource-constrained nodes [37].

6. Applicability in limited resource devices

One million transactions or so are recorded daily on Bitcoin; other BC networks have lower or higher transaction counts [1]. The amount of data on BC networks, such as Ethereum and Bitcoin, has grown significantly. The size of the BC for Bitcoin is about 350 GB as of October 2023, and it grows by about 1GB per day [1]. In contrast, the BC for Ethereum is more than 1.8 TB and is expanding faster than Bitcoin since it supports smart contracts. On the other hand, IoT devices come in a wide range of storage capacities: high-end devices can have up to 32 GB of storage, while ordinary edge devices range from 128 MB to 2 GB. Fog servers may store several terabytes to tens of terabytes of data. The landscape of data creation worldwide is proliferating; it is estimated that by 2025, IoT devices will produce 180 zettabytes of data annually [1]. Therefore, the volume of data produced by BC-based platforms like Bitcoin and Ethereum significantly outpaces the storage capacity of individual IoT and fog computing devices. Nevertheless, these devices may still communicate with BC networks by effectively managing and storing pertinent data via the use of techniques like pruning and BC

Table 4
Applicability of proposed techniques for resource-constrained devices.

Technique	Applicability	Concerns	Study
Structural modification	Limited	Changing consensus mechanisms and hard forking may compromise device performance and storage; however, SegWit can be advantageous to resource-constrained devices.	[36, 46]
Sharding	Applicable	Data loss and security need to be carefully considered.	[27, 40]
Ephemeral	Applicable	Data retention requirements, synchronization challenges, and security implications need to be considered.	[45]
Zk-SNARK	Limited	Applicability is determined by the security and privacy needs of the particular use case.	[27]
Pruning	Limited	Trade-offs between data retention, security, and resource constraints need to be carefully considered.	[34, 38]
Off-chain	Limited	Trade-offs between improved efficiency and the potential loss of decentralization and trustlessness.	[46]
Historical data	Applicable	The particular use case, data needs, and device resources all affect applicability.	[34]
Light clients	Applicable	Trade-offs between the benefits of BC and low functionality, trust, and security.	[35, 49]

compression, as discussed in Section 4.

This section answers RQ3 (the applicability of the techniques developed to address the BC bloat issue on resource-limited devices). While these techniques can be useful for minimizing BC bloat, their applications in resource-constrained devices like IoT and fog computing are restricted by several considerations. Table 4 summarizes these potentials and concerns regarding each of the seven techniques.

On the one hand, changing the consensus mechanism on limited-resource devices can be difficult. Modifying the consensus mechanism should thus be addressed with caution to prevent affecting the device's performance and to ensure that the adjustments do not overburden these devices with complicated computing chores. Implementing a hard fork on resource-constrained devices might be difficult since considerable changes to the software operating on these devices may be required [46]. Also, due to various considerations and restrictions, pruning, sharding, zk-SNARK, and off-chain techniques may have extremely limited use in resource-constrained devices.

Implementing pruning on devices with restricted resources necessitates a careful evaluation of the trade-offs between data preservation, security, and resource limits. In the case of sharding, especially concerning data loss and security in restricted resource devices, it is crucial to thoroughly consider the drawbacks and limitations [40]. To address these issues, selective use of sharding and hybrid approaches can be employed, tailoring solutions to specific use cases. However, zk-SNARKs introduce challenges related to scalability and computational complexity [27]. The suitability of zk-SNARKs for devices with limited resources is determined by the specific security and privacy requirements of the use case. In situations where data privacy and integrity are paramount, zk-SNARKs can be a valuable tool, provided that the computational overhead and proof size align with the constraints on the devices' resources [27]. Furthermore, off-chain solutions raise concerns about centralization and security. Their application on devices with limited resources is determined by specific use cases and the trade-offs between increased efficiency and the potential loss of decentralization and confidence. Off-chain solutions prove most beneficial in scenarios involving frequent and low-value transactions, allowing for the management or avoidance of centralization's boundaries. The appropriateness of these solutions is contingent on the particular needs and constraints of the devices in question.

Nevertheless, for devices with limited resources, alternatives such as SegWit, ephemeral, historical data, and light client solutions might be more fitting [36]. SegWit, for instance, reduces transaction size, subsequently lowering the demand for data storage and bandwidth, which proves advantageous for resource-constrained devices [46]. While ephemeral BC solutions are not universally adopted, they can be valuable in use cases prioritizing real-time processing, resource-constrained devices, and data reduction [45]. Implementers should carefully consider data preservation constraints, synchronization issues, and security concerns. Hybrid models that balance data preservation and real-time processing requirements may offer more adaptable solutions.

Historical data solutions, when implemented cautiously, can apply to resource-constrained devices. Though challenges related to data volume, storage costs, and privacy concerns exist, these issues can be mitigated through selective storage, edge solutions, local storage, adherence to data retention regulations, and the use of private or permissioned BCs. The usability of historical data solutions depends on the individual use case, data requirements, and resource limits of the network's devices and nodes [34]. Lastly, light client solutions, despite their limitations, such as centralization issues and security concerns, can be effectively employed in cases where reduced functionality and lower resource usage are acceptable [49]. However, its suitability is determined by the unique use case and the level of trust and security required.

7. Discussion and future research directions

BC is an emerging technology. While BC bloat has surfaced as a concern in recent years, solutions to this serious issue have recently been proposed. This study investigated these solutions, their limitations, and their usefulness in the setting of resource-constrained devices. Each technique has its own set of trade-offs and ramifications that should be considered when implementing them. To sum up, block structural modification necessitates a significant investment of time and effort. Sharding raises concerns about data loss and its security implications. Sharding demands intricate algorithms for shard communication. Ephemeral BCs pose synchronization and security risks. zk-SNARKs require specialized expertise for development and evaluation. Pruning may lead to permanent data loss, impacting historical data integrity and security. Off-chain storage methods face centralization challenges and latency issues. Historical data storage struggles with managing large volumes, impacting network performance. Light clients lack autonomous validation capabilities for transactions, smart contracts, or blocks.

Which technique to apply is determined by the unique use case, the necessity of data retention, and the available resources and restrictions. Addressing the BC bloat issue is critical to a cryptocurrency network's general acceptance and enhanced capacity. Both on-chain and off-chain technologies assist in retaining the integrity of the underlying BC while increasing transaction capacity significantly. However, the security of a particular BC, or even the integrity of the entire project, might be compromised by inherent risks. It will take time for big crypto-BC networks to improve bloat solutions. The most likely scenario is that on-chain solutions would prioritize security while allowing off-chain providers to adapt their offerings in response to certain situations [9].

Because there is a scarcity of scholarly literature, written in English, on approaches for dealing with BC bloat, we rely heavily on information accessible on professional websites. When performing comprehensive literature review research, this reliance on web-based sources is a constraint. While the web delivers significant insights, its information is not necessarily as carefully peer-reviewed, consistent, and academically validated as conventional scholarly sources. Despite this limitation, we want to shed light on this crucial problem and contribute to current knowledge in the field of BC bloat control.

8. Conclusions

As the quantity of data stored on BC networks increases, issues with data size, storage requirements, and operational inefficiencies occur. BC technology is continuously changing, with various techniques offered to handle the essential issue of BC bloat. As the volume of data on BC networks grows, these techniques provide critical data management and resource optimization options. This study identified and discussed eight techniques: structure modification, pruning, sharding, ephemeral, zk-SNARK, off-chain, historical data storage, and light client. While specific techniques, such as ephemeral and light client solutions, are more suited for resource-constrained devices, approaches like sharding and pruning can also be used, albeit with special prerequisites and limits relating to data management and security. The technique chosen is determined by the specific demands and constraints of the particular BC context. Knowing how each method matches these constraints is critical for effective implementation. Researchers and practitioners may pave the path for a more efficient and safe BC future by staying up-to-date on the newest advancements and solutions. Furthermore, future research may consider developing frameworks or algorithms that autonomously select the most suitable BC techniques based on the specific demands and constraints of the context. This could involve machine learning or context-aware decision-making processes.

CRediT authorship contribution statement

Yehia Ibrahim Alzoubi: Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Alok Mishra:** Methodology, Formal analysis, Validation, Writing – review & editing.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

References

[1] Statista. Blockchain - statistics & facts. https://www.statista.com/topics/5122/blockchain/, accessed 4 January 2024. 2024.
[2] Abbasi M, Prieto J, Shahraki A, Corchado JM. Industrial data monetization: a blockchain-based industrial IoT data trading system. Internet Things 2023;24:100959.
[3] Akrasi-Mensah NK, Tchao ET, Sikora A, Agbemenu AS, Nunoo-Mensah H, Ahmed A-R, Welte D, Keelson E. An overview of technologies for improving storage efficiency in blockchain-based IIoT applications. Electronics 2022;11:2513.
[4] Khalid MI, Ehsan I, Al-Ani AK, Iqbal J, Hussain S, Ullah SS. A comprehensive survey on blockchain-based decentralized storage networks. IEEE Access 2023;11:10995–1015.
[5] Davies, A. Can blockchain bloat ever be solved? https://www.devteam.space/blog/can-blockchain-bloat-ever-be-solved/, accessed 12 October 2023. 2022.
[6] Alzoubi YI, Mishra A. Green blockchain—A move towards sustainability. J Clean Prod 2023. https://doi.org/10.1016/j.jclepro.2023.139541.
[7] Bitcoinist. Blockchain data bloat in the supply chain. https://bitcoinist.com/ephemeral-blockchains-fix-data-bloat/, accessed 12 October 2023. 2022.
[8] Zhou K, Wang C, Wang X, Chen S, Cheng H. A novel scheme to improve the scalability of bitcoin combining IPFS with block compression. IEEE Trans Netw Serv Manag 2022;19:3694–705.
[9] Wade, J.; Silberstein, S.; Velasquez, V. Layer 1 vs. layer 2: the difference between blockchain scaling solutions. https://www.investopedia.com/what-are-layer-1-and-layer-2-blockchain-scaling-solutions-7104877, accessed 23 October 2023. 2023.
[10] Mahmud M, Sohan MSH, Reno S, Sikder MB, Hossain FS. Advancements in scalability of blockchain infrastructure through IPFS and dual blockchain methodology. J Supercomput 2023:1–23. https://doi.org/10.1007/s11227-023-05734-x.
[11] Zhang Q, Zhao Z. Distributed storage scheme for encryption speech data based on blockchain and IPFS. J Supercomput 2023;79:897–923.
[12] Li Z, Li G, Bilal M, Liu D, Huang T, Xu X. Blockchain-assisted server placement with elitist preserved genetic algorithm in edge computing. IEEE Internet Things J 2023;10:21401–9.
[13] Hong Z, Guo S, Zhou E, Chen W, Huang H, Zomaya A. GriDB: scaling blockchain database via sharding and off-chain cross-shard mechanism. Proc VLDB Endow 2023;16:1685–98.
[14] Pawar V, Sachdeva S. ParallelChain: a scalable healthcare framework with low-energy consumption using blockchain. Int Trans Oper Res 2023:1–29. https://doi.org/10.1111/itor.13278.
[15] Ren Y, Liu X, Sharma PK, Alfarraj O, Tolba A, Wang S, Wang J. Data storage mechanism of industrial IoT based on LRC sharding blockchain. Sci Rep 2023;13:2746.
[16] Gurunathan M, Mahmoud MA, Faisal FH. Enhanced blockchain scalability for IoT-based smart devices-a generic model development. In: Proceedings of the 13th symposium on computer applications & industrial electronics (ISCAIE). IEEE; 2023. p. 320–5.
[17] Martinez KKC. Blockchain scalability solved via quintessential parallel multiprocessor. In: Proceedings of the 2023 international wireless communications and mobile computing (IWCMC). IEEE; 2023. p. 1626–31.
[18] Capretto M, Ceresa M, Fernández Anta A, Russo A, Sánchez C. Improving blockchain scalability with the setchain data-type. Distrib Ledger Technol: Res Pract 2023. https://doi.org/10.1145/3626963.
[19] Chorey P, Sahu N. Enhancing efficiency and scalability in blockchain consensus algorithms: the role of checkpoint approach. J Integr Sci Technol 2024;12:706.
[20] Zhen Z, Wang X, Lin H, Garg S, Kumar P, Hossain MS. A dynamic state sharding blockchain architecture for scalable and secure crowdsourcing systems. J Netw Comput Appl 2024;222:103785.
[21] Davies J. Enhanced scalability and privacy for blockchain data using Merklized transactions. Front Blockchain 2024;6:1222614.
[22] Hafid A, Hafid AS, Samih M. A tractable probabilistic approach to analyze sybil attacks in sharding-based blockchain protocols. IEEE Trans Emerg Top Comput 2022;11:126–36.

[23] Gracy M, Balasundaram RJ, Raj SAA. Enhancing data integrity in blockchain through fuzzy augmented lagrangian optimization and compact blocks to minimize redundancy. Int J Intell Syst Appl Eng 2023;11:387–401.
[24] Kuznetsova K, Yezhov A, Kuznetsov O, Tikhonov A. Solving blockchain scalability problem using zK-SNARK. In: Hu Z, Zhang Q, He M, editors. Advances in artificial systems for logistics engineering III. Lecture notes on data engineering and communications technologies, 180. Cham: Springer; 2023. p. 360–71.
[25] Peng S, Bao W, Liu H, Xiao X, Shang J, Han L, Wang S, Xie X, Xu Y. A peer-to-peer file storage and sharing system based on consortium blockchain. Future Gener Comput Syst 2023;141:197–204.
[26] Thanalakshmi P, Rishikhesh A, Marion Marceline J, Joshi GP, Cho W. A quantum-resistant blockchain system: a comparative analysis. Mathematics 2023;11:3947.
[27] Bespalov Y, Kovalchuk L, Nelasa H, Oliynykov R, Viglione R. Models for generation of proof forest in zk-SNARK based sidechains. Cryptography 2023;7:14.
[28] Tsang Y, Lee C, Zhang K, Wu C, Ip W. On-chain and off-chain data management for blockchain-internet of things: a multi-agent deep reinforcement learning approach. J Grid Comput 2024;22:1–22.
[29] Huang J, Kong L, Wang J, Chen G, Gao J, Huang G, Khan MK. Secure data sharing over vehicular networks based on multi-sharding blockchain. ACM Trans Sensor Netw 2024;20:1–23.
[30] Khan D, Jung LT, Hashmani MA. Systematic literature review of challenges in blockchain scalability. Appl Sci 2021;11:9372.
[31] Fan X, Niu B, Liu Z. Scalable blockchain storage systems: research progress and models. Computing 2022;104:1497–524.
[32] Liu Y, Fang Z, Cheung MH, Cai W, Huang J. Mechanisms design for blockchain storage sustainability. IEEE Commun Mag 2023;61:102–7.
[33] Hashim F, Shuaib K, Zaki N. Sharding for scalable blockchain networks. SN Comput Sci 2022;4:1–17.
[34] Liu Y, Liu J, Salles MAV, Zhang Z, Li T, Hu B, Henglein F, Lu R. Building blocks of sharding blockchain systems: concepts, approaches, and open problems. Comput Sci Rev 2022;46:100513.
[35] Singh BC, Ye Q, Hu H, Xiao B. Efficient and lightweight indexing approach for multi-dimensional historical data in blockchain. Future Gener Comput Syst 2023;139:210–23.
[36] Kedziora M, Pieprzka D, Jozwiak I, Liu Y, Song H. Analysis of segregated witness implementation for increasing efficiency and security of the Bitcoin cryptocurrency. J Inf Telecommun 2023;7:44–55.
[37] Yadav B, Gupta S. Comparative cost analysis of on-chain and off-chain immutable data storage using blockchain for healthcare data. In: Choudrie J, Mahalle P, Perumal T, Joshi A, editors. IOT with smart systems. Smart innovation, systems and technologies, 312. Singapore: Springer; 2022. p. 779–87.
[38] Bowlin EW, Khan MS, Bajracharya B. Reducing bootstrap overhead within VANET blockchain applications through pruning. In: Proceedings of the international conference on smart applications, communications and networking. IEEE; 2023. p. 1–6.
[39] Gong F, Kong L, Lu Y, Qian J, Min X. An overview of blockchain scalability for storage. In: Proceedings of the 26th international conference on computer supported cooperative work in design. IEEE; 2023. p. 516–21.
[40] Alshahrani H, Islam N, Syed D, Sulaiman A, Al Reshan MS, Rajab K, Shaikh A, Shuja-Uddin J, Soomro A. Sustainability in blockchain: a systematic literature review on scalability and power consumption issues. Energies 2023;16:1510.
[41] Rao IS, Kiah M, Hameed MM, Memon ZA. Scalability of blockchain: a comprehensive review and future research direction. Clust Comput 2024:1–24. https://doi.org/10.1007/s10586-023-04257-7.
[42] Heo JW, Ramachandran GS, Dorri A, Jurdak R. Blockchain data storage optimisations: a comprehensive survey. ACM Comput Surv 2024:1–26. https://doi.org/10.1145/3645104.
[43] Liu Z, Hu C, Xia H, Xiang T, Wang B, Chen J. SPDTS: a differential privacy-based blockchain scheme for secure power data trading. IEEE Trans Netw Serv Manag 2022;19:5196–207.
[44] OKX. Crypto for everyone. https://www.okx.com/learn, accessed 1 February 2024. 2024.
[45] Frąckiewicz, M. Ephemeral computing and blockchain: a synergy for enhanced security. https://ts2.space/en/ephemeral-computing-and-blockchain-a-synergy-for-enhanced-security/, accessed 12 October 2023. 2023.
[46] Blockchain. What is SegWit and its benefits? https://support.blockchain.com/hc/en-us/articles/4417071701140-What-is-SegWit-and-its-benefits, accessed 29 October 2023. 2023.
[47] Staff, P. Bitcoin Core developer proposes new type of pruned node. https://protos.com/bitcoin-core-developer-proposes-new-type-of-pruned-node/, accessed 26 October 2023. 2023.
[48] Dwyer, K. What are cryptocurrency layer 2 scaling solutions? https://coinmarketcap.com/academy/article/what-are-cryptocurrency-layer-2-scaling-solutions, accessed 22 October 2023. 2022.
[49] Nervos. What is a light client in blockchain technology? https://www.nervos.org/knowledge-base/what_is_a_light_client_(explainCKBot), accessed 30 October 2023. 2023.
[50] Alzoubi YI, Gill A, Mishra A. A systematic review of the purposes of blockchain and fog computing integration: classification and open issues. J Cloud Comput 2022;11:1–36.