Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.036079

ARTICLE

Hash Table Assisted Efficient File Level De-Duplication Scheme in SD-IoV Assisted Sensing Devices

Ghawar Said[1], Ata Ullah[2], Anwar Ghani[1,*], Muhammad Azeem[1], Khalid Yahya[3], Muhammad Bilal[4] and Sayed Chhattan Shah[5,*]

[1]Department of Computer Science, International Islamic University, Islamabad, 44000, Pakistan

[2]Department of Computer Science, National University of Modern Languages, Islamabad, 44000, Pakistan

[3]Department of Electrical and Electronics Engineering, Nisantasi University, Istanbul, 34467, Turkey

[4]Department of Computer Engineering, Hankuk University of Foreign Studies, Yongin-si, Gyeonggi-do, 17035, Korea

[5]Department of Information and Communication Engineering, Hankuk University of Foreign Studies, Yongin-si, Gyeonggi-do, 17035, Korea

*Corresponding Authors: Anwar Ghani. Email: anwar.ghanir@iiu.edu.pk; Sayed Chhattan Shah. Email: shah@hufs.ac.kr

Received: 16 September 2022 Accepted: 02 March 2023 Published: 26 January 2024

ABSTRACT

The Internet of Things (IoT) and cloud technologies have encouraged massive data storage at central repositories. Software-defined networks (SDN) support the processing of data and restrict the transmission of duplicate values. It is necessary to use a data de-duplication mechanism to reduce communication costs and storage overhead. Existing State of the art schemes suffer from computational overhead due to deterministic or random tree-based tags generation which further increases as the file size grows. This paper presents an efficient file-level de-duplication scheme (EFDS) where the cost of creating tags is reduced by employing a hash table with key-value pair for each block of the file. Further, an algorithm for hash table-based duplicate block identification and storage (HDBIS) is presented based on fingerprints that maintain a linked list of similar duplicate blocks on the same index. Hash tables normally have a consistent time complexity for lookup, generating, and deleting stored data regardless of the input size. The experiential results show that the proposed EFDS scheme performs better compared to its counterparts.

KEYWORDS

Hash table; de-duplication; linked list; IoT; sensing devices

1 Introduction

The Internet of Things (IoT) is a conceptual network of things, machines, or objects and devices. Sensors and connectivity of the network enable these things to gather and exchange data [1]. It is a speedy rising network to transform human life by significantly increasing its applicability in Healthcare, Agriculture, Smart Traffic Management, Smart Home/City, Security Surveillance, Industrial Internet, and Wearable [2–5]. IoT is a new paradigm/model, which agrees with a large number of smart things that can be associated with the Internet. These object devices like actuators and sensors are capable of managing and exchanging that data for further processing without any human interference. The transition from traditional Vehicle Ad-hoc Networks to the Internet of Vehicles is being driven by the new Internet of Things age (IoV). IOV promises enormous commercial interest and research value, drawing a significant number of businesses and academics as a result of the quick growth of computation and communication technologies. In the IoT and IoV, the role of smart sensing devices is significant, as in many places the deployment of wired configuration is not possible easily [6].

A vast number of sensor nodes work together to send data to the sink or collector node like the roadside units. Sensor nodes are responsible for sensing and collecting information from their surroundings. A small battery that, in most situations, cannot be replaced powers these nodes [7]. The task of sensing devices in IoT is to sense data from those areas where it is deployed and then communicate that data to the central controller for further processing [8]. An emerging idea to realize the promise of intelligent transportation systems (ITSs) is the Internet of Vehicles (IoV). High throughput satellite communication, the IoT, and cyber-physical systems are the researcher's current research areas as a result of the rapid advancements in automotive technology [9]. Software-Defined Networking (SDN) based IoV enhances the capability to filter out the unnecessary data being transmitted on the network in the existing setup. The SD-IoV is an advantageous combination for designing mobile edge computing-based IoV techniques. In this context, the performance and quality of services of the IoV system can be enhanced by interacting with SDN and Network Function Virtualization (NFV). Moreover, SD-IoV-based applications still face performance and scalability issues. Thus, the transmission and computational perspectives in the SD-IoV are still challenging issues for future research studies. Data redundancy is a serious problem that wastes a lot of storage capacity in setups with integrated cloud and fog storage.

The de-duplication process helps to reduce or remove the duplicate data. It preserves the unique ones in such a way that the scheme may fully retrieve the data at any moment [10]. The SD-IoV involves massive amount of data being processed at the SDN controller which enhances the use of de-duplication to reduce the communication cost by eliminating the redundant data. The SD-IoV can restrict the transmission of certain redundant values like temperature, blood pressure, heart rate of the driver etc. and the break oil, gear oil and Mobil oil levels from the collector nodes. There may be a condition of sending the values when there is a change or just send a small Boolean indicator to ensure that there is no change in the previously saved values. The SDN based IoV enhances the capability to filter out the unnecessary data being transmitted on the network in the existing setup. The SD-IoV is an advantageous combination for designing mobile edge computing-based IoV techniques. In this context, the performance and quality of services of the IoV system can be enhanced by interacting with SDN and Network Function Virtualization (NFV). SDN enables the creation of network-aware applications, intelligent network state monitoring, and automatic network configuration adaptation.

Various data compression strategies have been explored in the article [11], including application-based, data type-based, and data coding-based. Smart compression or de-duplication, in contrast to other compression algorithms, removes duplicate data from numerous files to minimize storage space. The data de-duplication technique is useful when a big volume of data is available [12]. Despite the rapid rise of cloud computing and large data, de-duplication has become a popular topic in recent years. The de-duplication procedure, which avoids storing the same data several times in real-time, lowers the cost of cloud storage significantly [13]. By data de-duplication, the number of average transmitted messages throughout the network which includes transmissions between peers at various hierarchies of the network, and transmissions from the sensor field to the sink node will be reduced which in turn decreases the power consumption. Bandwidth is also resource-constrained in sensor networks, De-duplication helps in the effective utilization of bandwidth by eliminating the duplicates predominantly. With the help of de-duplication, the network traffic predominantly decreases, and consequently, data loss due to collisions is also decreased [14,15]. Smart sensor-based Networks involve a large number of smart sensors that aim to collect data movement, process, analyze, and transfer information [16]. For the enhancement of existing issues, the proposed scheme EFDS uses the novel HDBIS algorithm, for the reduction of encryption, decryption time, block generation, data de-duplication time, block data rate, and block storage time.

The motivation of the article is to enhance different parameters like encryption, decryption, block generation, and data de-duplication time. Several existing schemes provide data de-duplication methods but still have different open challenging issues that should be overcome in future research concerns. A quick searching mechanism is needed to manage data packets and the network's quality. A hash table, which keeps data in an array and uses it to determine where elements should be added or located, uses a hash function to build an index. The cost of running a hash function needs to be lower for employing the hashing strategy to be preferable to other traditional methods. Therefore, the article presents Link-list and Hash Table methods to enhance the above parameters. The main contribution of this work is tabulated as follows:

1. To propose a scheme for managing the duplicate blocks and storing them as a linked list within a hash table to reduce the computational overhead of tree-based tag creation.
2. To design an algorithm (HDBIS) that can reduce the block generation time, data de-duplication time, and operation time.
3. To comparatively analyze the proposed EFDS scheme with counterparts in terms of performance and investigate enhancement in the data de-duplication, and the block storage time.

The rest of the article is structured as follows. Section 2 discusses the literature review. The system model and identification of the problem are presented in Section 3. Section 4 outlines the proposed Scheme. The results have been discussed and analyzed in Section 5 and Section 6 concludes the work.

2 Literature Review

This section explores the data de-duplications schemes for IoT-enabled sensing devices along with support for SD-IoV scenarios. This paper analyzes the data de-duplication scheme for file-level data de-duplication schemes with different methods. Several schemes involve sensing devices to send data to central repositories. In [17], the majority of distributed applications and cloud services including Bit Torrent, Google Cloud, and AWS rely on hashing techniques that concur with a healthy and effective hash table's dynamic scaling. For instance, as the hash database grows, reliable and engagement-hashing techniques reduce key remapping. The key feature that disturbs the search process to search and recover information efficiently is the way the data is arranged [18]. Due to the wide variety of storage devices and the enormous growth in the amount of data coming from many sources today, such as commercial transactions, social networks, and many other areas, searching is one of the key subjects in computer science. For finding an element in a set of items in a hash table T, hashing is one of the effective data retrieval strategies [19].

In [20], two secure data de-duplication schemes based on Rabin fingerprinting (SDRF) using deterministic and random tags are presented. The de-duplication is enabled before the data is outsourced to the cloud storage server. In particular, the scheme used the variable-size block-level de-duplication using Rabin fingerprinting. However, the weak points of the de-duplication time depend on the sliding step size of the window and the block generation time is more than the fixed-size block scheme. In [21], the authors proposed the P-Dedupe: Exploiting Parallelism in data de-duplication system, which in turn uses content-defined chunking (CDC) during de-duplication. P-Dedupe first pipelines the four stages of the de-duplication tasks with the processing units of chunk files and then further processed in parallel for CDC. The main issue of P-Dedupe is to decrease the deduplication ratio.

In [22] a Smart De-duplication scheme for Mobiles (SDM) is proposed which is a better de-duplication method without specific configurations for any file type of data. The SDM scheme is suitable for mobile devices only. In [23], a secure Healthcare data (SHD) sharing scheme in a Joint-Cloud storage arrangement delivers global services via association with several clouds. SHD also supports dynamic data information and sharing without the help of the trusted KS. The SHD method is only efficient for mobile devices. In [24], the Asymmetric Extremum (AE) algorithm was proposed that uses variable-sized and fixed-sized windows on the left and right-hand sides. The extremely valued byte is fitted in the middle of the two windows where AE scans each byte to find the cut-point. The main issue in this scheme is that the size of the variable sized-window will be zero bytes when the leading bytes are true for the condition. As it compares each byte to the fixed-sized window so an account of this computation cost increase. In [25], a novel three-tier secure data de-duplication architecture is condensed as SEEDDUP for data storage and recovery in the cloud. SEEDDUP delivers duplicates with appropriate indexing and the privacy of the user. Using the "Cuckoo Search Algorithm", the verification of chunk size in the cloud is verified by the hash value generated by the SHA3-512. The weak point of it is to consume more time for the preprocessor.

The authors in [26] investigated the de-duplication threshold in cloud storage and suggested a secure de-duplication strategy for IoT sensor networks using a dynamic threshold adjustment. In [27], a CDC scheme with a new chunking algorithm called Rapid Asymmetric-Maximum (RAM) is proposed. The scheme works by searching a byte with a large value in the fixed sized-window. The main issue arises in some cases where the index value is greater than the fixed window size, the size of a variable-sized window can be very small. In [28], the "Secure and efficient big data de-duplication in fog computing (SED)" shows a new decentralized de-duplication structure before demonstrating how to use it to build a secure and efficient massive data de-duplication strategy in fog computing. The computation cost of SED is speedily increased when the size of the data is large. A bucket based-technique that uses buckets to store data with a Fixed Size Chunking algorithm for chunk generation based on the MAD5 Algorithm to generate the hash values is presented in [29]. A Frequency-based scheme with high de-duplication capability is presented in [30] where FSC and CDC chunking is used. The simplest and fastest approach is FSC which breaks the input stream into fixed-size chunks. Since the scheme uses only fixed-size chunks, therefore, the frequency estimation phase takes a long time for data, thereby consuming more energy during computations.

An efficient de-duplication approach for cluster de-duplication system ARD-edup to achieve high data de-duplication rate, low communication overhead, and manage load balancing is presented in [31]. As data is growing rapidly in data centers, the inline cluster de-duplication technique has been widely used to improve storage efficiency and data reliability. The weak point of it is long time consumption when load imbalance occurs for extreme binding. In [32], a Dynamic de-duplication approach is proposed for big-data storage that explores a dynamic de-duplication approach with greater efficiency using CDC, static-chunking, delta encoding, and whole file chunking algorithms. The weak point of it is consuming more time for the preprocessor. In [33], two advanced energy-efficient architecture systems are presented to enhance the lifetime of sensing devices used in mobile healthcare applications applicable for IoV as well. In [34], the authors proposed a new way for constructing safe de-duplication systems in the cloud and fog using Convergent and Modified Elliptic

Curve Cryptography (MECC) algorithms. For MECC algorithms, the time it takes to generate a hash tree and tags is extremely long, and as the number of uploaded files grows higher, the rate of increase rises faster and faster, with a steep upward trend. The procedure for generating tags is exceedingly difficult. Data redundancy needs to be minimized on the one hand, and a strong encryption strategy needs to be devised on the other to assure the security of the data. This method works well for tasks like a user uploading new files to cloud storage or the fog.

In [35], authors used a combination of two new approaches for chunking and hashing algorithms. The first method is Bytes Frequency-Based Chunking (BFBC), which uses the frequency of the data-set byte occurrence information to improve data de-duplication gain. While the method used is the triple hash technique, which employs a mathematical function to generate short fingerprints and has a direct impact on index table size and hashing throughput. The FBC scheme is to apply only fixed chunks and the frequency estimation phase takes a long time to calculate the data and more energy consumption during computations.

In [36], a secure and distributed outsourcing algorithm proposed for modular exponentiation (fixed base and variable exponent) under the multiple non-colluding edge node model. In addition, this work proposed another secure and distributed outsourcing algorithm of modular exponentiation (variable base and variable exponent). In [37], an "Edge facilitated de-duplication technique, EF-dedup" partitions sub-edge nodes into self-governing de-duplication clusters (referred to as rings). It carefully balances the de-duplication ratio and throughput. In [38], it presented "an effective radix trie (RT) with Bloom Filter (BF) based secure data de-duplication model", condensed as SDD-RT-BD. The presented SDD-RT-BF model involves three main steps namely, proof of ownership, authorized de-duplication, and role key update. For Memcached applications, Yang et al. [39] proposed a pipelined hash table architecture that can support up to 10 Gbps throughput. A hash table designed for networking applications by Tong et al. [40] can handle up to 85 Gbps of throughput. In [41], the SDN approach enables the sharing of a medium between Electronic-Control-Modules (ECUs) and in-vehicle data sources. SDN greatly improves the performance of the IoV-based application by separately dealing with application controls and application data so the centralized control approach provides a broad picture of the network for efficient network optimization. Therefore, the SDN provides efficient data computation, communication, and less storage utilization. In [42], flexible architecture for car Ethernet networks is presented that could coexist with both time-sensitive low-bandwidth data and multimedia streams with high bandwidth. The in-vehicle network can be reconfigured to accommodate new hardware and applications based on the SDN paradigm. There is a challenging issue to design a novel SDN system that provides support in several fields with different scales of an SDN controller.

An inline deduplication approach for persistent memory PM, known as LO-Dedup, is presented in [43] and can be used for IoT devices like sensors. The time it takes to calculate a two-level fingerprint, with the first fingerprint using a partial hash and the second using MurmrHash3, is a drawback of the current technique. Second, it takes a very long time to generate hash tree tags using a PM-friendly index structure and tree structure, and when big-size files get bigger, the rate of increase climbs steeply upward.

3 Problem Identification and System Model

Sensor nodes send large amounts of data continuously to the SDN controllers in the SD-IoV scenario. The SDN controller transfers the redundant data to the cloud repositories. Large storage space wasted by saving the same data on the cloud servers in SD-IoV repeatedly. In the existing scheme, the tag generation for the data de-duplication scheme SDRF [20] uses deterministic tags and random tags. The tag generation of both types is time expensive, which increases gradually with the size of the files to be uploaded. The process for the generation of tags is very complex and not efficient and increases the computational cost with the complicated method of tags generation. In the existing scheme of MECC [34], the hash tree and its tags generation time are extremely long. Moreover, as the number of uploaded files grows higher, the rate of increase rises faster and faster, with a steep upward trend. The procedure for generating tags is exceedingly extensive which increases computing and communication costs. Centralized SDN controllers are introduced to greatly improve communication. Fig. 1 illustrates the system model, which consists of three entities: SDN Controller, Sensor nodes, and cloud storage server (CSS) in SD-IoV. According to the system model, the sensor nodes outsource their data to SDN Controller for data de-duplication, which creates a hash table for all the received sub-blocks to assign tags and identify the duplicate blocks. SDN controller finally generates the de-duplicated list to share with the cloud.

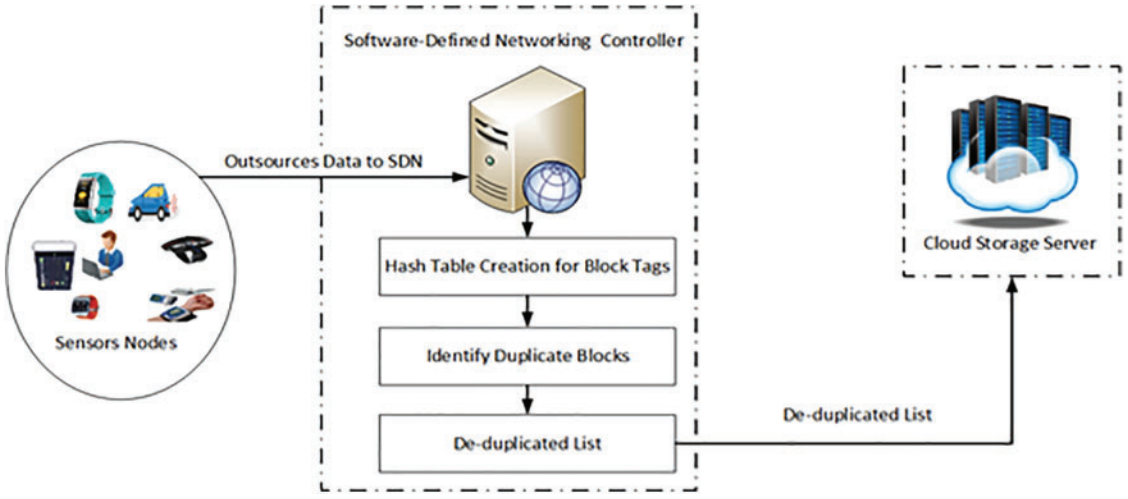
Software-Defined Networking Controller
Outsources Data to SDN
Sensors Nodes
Hash Table Creation for Block Tags
Identify Duplicate Blocks
De-duplicated List
De-duplicated List
Cloud Storage Server

Figure 1: System model

Hash Table is a data structure that stores data in an associative way. A hash function creates an index in a hash table, which stores data in an array and uses it to identify where elements should be added or located. To make using the hashing approach superior to another conventional way, the cost of running a hash function must be lower. Hash tables, which have a linear time complexity, and binary search trees, which have a logarithmic time complexity, are often more effective at seeking up values than search trees. Hash tables normally have a consistent time complexity for looking up, generating, and deleting stored data regardless of the input size. In the SDN Controller, identify duplicated blocks through the hash table techniques. After that, maintain the link list of duplicated blocks for further processing. A linked list is a linear data arrangement that dynamically stores data elements. The size of the linked list can be increased or decreased at run time so there is no memory wastage.

4 Proposed Efficient File Level De-Duplication Scheme (EFDS)

To address the identified existing problems, this paper presents the proposed solution for the file-level de-duplication mechanism. It uses a hash-table-based mechanism to identify the duplicate blocks from the file shared by the sensing devices toward the SDN controllers. The set of multiple sensing devices involves a collector or head node to send an aggregated data file. The sensor devices send large amounts of data continues to edge Servers in redundant data. In each scenario, the sensor devices on the vehicles upload a data file to a cloud storage server through the SDN Controller. This paper presents a novel algorithm executed at the SDN Controller, which divides the sensing data file into sub-blocks. SDN controller creates a hash table for all the received sub-blocks and maintains a linked list for the duplicate blocks on the same index generated through hash based fingerprinting. A list of notations is shown in Table 1.

Table 1: List of notations

Symbol	Description
LLD	Linked list of duplicate blocks
HTBL	Hash table for blocks of a file
CSS	Cloud storage server
T_Size	Total size of data
M	Index number
C^*	Generate cipher text
B[ ]	An array of blocks as strings
K^*	Convergent key
H(Bi)	Hash of block Bi
F[m]	A fingerprint of block Bi

In Fig. 2 this paper presents the visual model for maintaining the hash table for the identified block extracted from the sensing data file. It shows the Rabin fingerprint as f(B_i) for each B_i. It also takes the hash for each B_i as H(B_i) for arranging values in the hash table. This paper has identified that the same index can generate in the hash table, which can either be resolved through adjusting the duplicate value in any next available slot but it causes primary clustering by gathering most of the values in a specific region where other regions of the table are left blank. Adopting the mechanism of quadratic-based index calculation can avoid the massive clustering at one place but it results in clustering at the distributed position as similar indices may be generated. Finally, separate chaining is adopted for duplicate blocks by maintaining the linked lists where the same index is generated by the hash function. A linked list is a linear data arrangement that stores a collection of data elements dynamically. The size of the linked list can be increased or decreased at run time so there is no memory wastage. It consumes very less computation time to search and arrange the blocks instead of maintaining the tree of tags in base schemes.

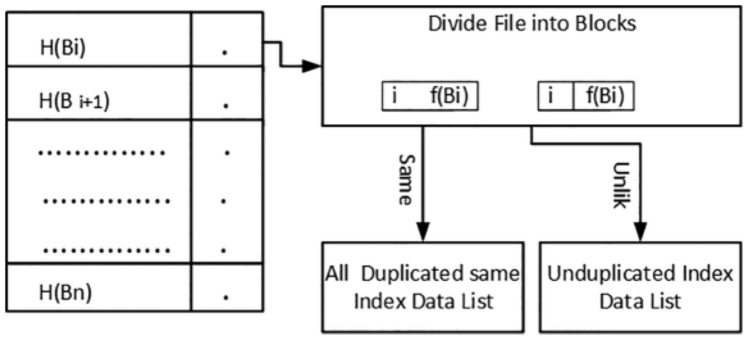

Figure 2: Hash table and fingerprint-based duplicate block segregation

The novel Hash-Table-based Duplicate Block Identification and Storage (HDBIS) Algorithm is presented based on fingerprints that establish a link list for identical duplicate blocks at the same index generated by the hash function. The detailed steps of the algorithm are as follow. Initially, a file is received from the collector node as input. The collector node may be the individual sensing device like a vehicle or the head/collector node to share the aggregated data file for member nodes. In aggregated data files, the chances of duplications are high like sharing the same location, speed, temperature, and other sensing conditions. At the beginning of the algorithmic steps, declare or initialize the values of T_size as −1, m as 0, B[] as an array of strings, and set LLD[] as a linked list for duplicate blocks to maintain a linked list for duplicate blocks. Next, the HTBL[] is declared for a hash table for the creation of blocks within a table size T_size. Moreover, a get file method is utilized to access the file and set the file_size of the received file. A hash method is utilized to establish an index of the hash table that stores data in an array. A hash table is helpful to identify where a new element is placed and also provides fast access to located specific elements. Thus, the hash table technique is utilized to determine the duplicated blocks. After that, a link list is maintained for duplicated blocks that generate the same hash and reach the same index inside the hash table. It must be stored dynamically by maintaining a chain in the form of a linked list. It will help to collect the duplicates in a sequence and either access or remove them. In steps 1–11, the time T_Size+1, means that $1 - 1 = 0$ as the T_zise is initialized with -1. In step 7, the while loop is applied with a condition of T_BlK_Size ≤ File Size to execute it unless the entire file is not processed. The substring of the file is created by adding the T_Size and BlK_size of the given file and saved in the array of index position m. The Rabin fingerprint is used to obtain *F[m]*. It looks up in the *HTBL[]* to access the index of the *F[m]*.

In this context, if the index value is null then there is no duplicate found for given F[m], and add the given block in the *HTBL[]* at another index. Otherwise, add the given block in the *LLD[]* at *HTBL[]* index and increment the table value with the block size. For example, suppose that block size BLK_size is 50 MB. From T_size 0 to 50 MB take over to array index position m = 0, as B[m] is saved as an index of zeroth address. Similarly, in the second iteration, the increment of the T_size by BLK_size means that the first time, have zero then add $0 + 50 = 50$ in BLK_size. Next loop iteration substituted as $50 + 1 = 51$. Like $50 + 50 = 100$, assign to B[m] means that Index m = 1 position data. From step 12 to step 14 consisting the SDN Controller generates a randomized convergent Key K^* with the user and CSS. After this, the SDN Controller generates a cipher text like $C^* = \text{Enc}\,[k^*, M] =$ [IDS, T S, HT B [], Hash [Ml]. Lastly, the SDN Controller Transmits the (IDSDN, C^*) toward CSS. Our presented algorithm provides optimized resource consumption by reducing the redundant data storage at cloud repositories.

Algorithm 1: Hash-table-based duplicate block identification and storage (HDBIS) algorithm

```
Input: Take a data file from the collector node
Output: Hash Table with identified duplicates in the linked list
1: Set T_Size to −1
2: Set m to 0
3: Set B[ ] as an Array of Strings
4: Set LLD[ ] as a linked list of duplicate blocks
5: Set HTBL[ ] as a Hash table for Blocks of a file
6: Set File_Size = File -> getSize( )
7: while T_Blk_Size ≤ File_Size do
8:    B[m] = File− > Substring(T_Size+1,T_Size+Blk_Size)
```

(Continued)

Algorithm 1 (continued)

```
9:  F[m] = generate Rabin-fingerprint(B[m])
10: Lookup HTBL for F[m]
11: Set index to get Index of (HTBL[]) for F[m]
12: if index != NULL then
13:     Add F[m] in LLD at HTBL[index]
14:   else
15:     Add F[m] in Hash table HTBL[ ]
16:   end if
17:   T_Size = T_Size+Blk_Size
18: end while
19: SDN Controller: Generate a randomized Convergent Key $K^*$ with user and CSS
20: SDN Controller: Generate Cipher $C^* = Enc\{K^*, M1= [IDS, TS, HTBL[]], Hash (M1)\}$
21: SDN Controller: Transmits the (IDSDN, $C^*$) toward CSS
```

5 Results and Analysis

This section explores the results for the proposed EFDS scheme in comparison to base schemes. A test bed is established by developing a server-side application using C# and ASP.net based web-application. Moreover, the Windows Communication Foundation (WCF) services are developed to implement the back-end logic for de-duplication and deployed on the Windows Azure cloud. The functions of the WCF service are remotely called from the web application. Moreover, the EFDS algorithm performance is compared with SDRF and MECC schemes. In this context, SDRF and MECC algorithms are also implemented to evaluate the performance of these algorithms as compared to EFDS on the same data file size. The repository for de-duplication data is maintained at SQL server 2019. A linked list is a linear data arrangement that stores a collection of data elements dynamically. The size of the linked list can be increased or decreased at run time so there is no memory wastage. It consumes very less computation time to search and arrange the blocks instead of maintaining the tree of tags in base schemes.

Furthermore, the de-duplication rate is evaluated with various file and block sizes. Moreover, block storage and generation time are also considered against different file sizes. Different performance evaluation parameters include encryption time, decryption time, de-duplication rate, block storage time, and block generation time. A list of simulation parameters is shown in Table 2.

Table 2: Simulation parameters

Parameters	Values
Number of files	50 files
Maximum file size	40 MB
Minimum file size	5 MB
Minimum block size	10 KB
Maximum block size	1 MB

5.1 Encryption Time

Fig. 3a, the EFDS encryption time is evaluated compared with counterparts. In the case of a file size of 15 MB, the EFDS, SDRF, and MECC schemes take 11.51, 13.38, and 17.46 s time for encryption of all blocks in for the entire file. The results show that the EFDS provides better for encryption time than SDRF and MECC. Table 3 shows a comparative analysis of the proposed EFDS scheme with the other counterparts for different sizes of data in MB. SDRF and MECC. For a case of 20 MB data size, the proposed EFDS, and the existing schemes; SDRF, and MECC require 13.29, 15.34, and 19.44 s encryption time, respectively.

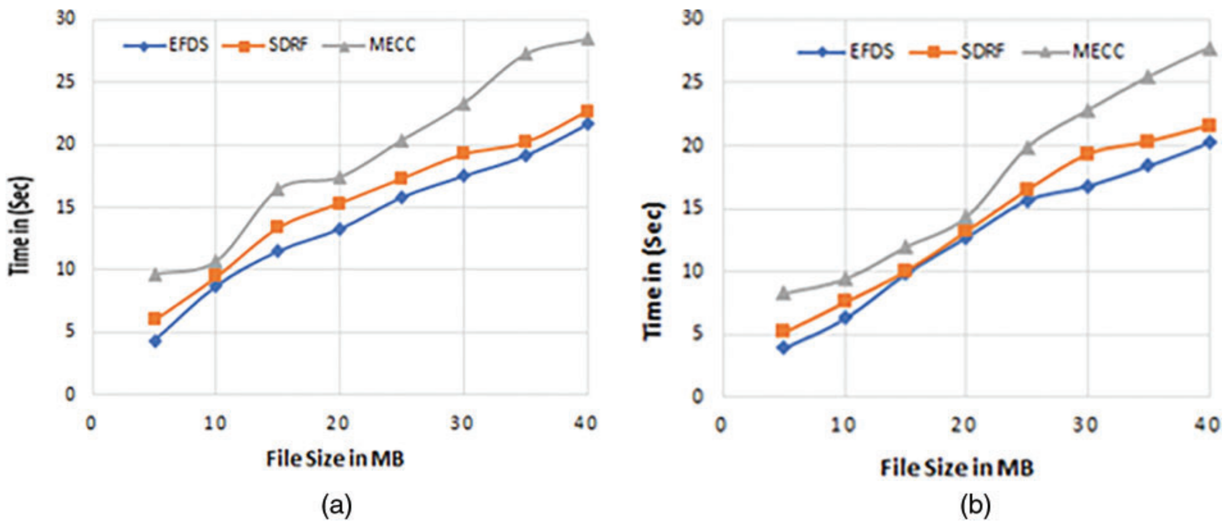

Figure 3: Encryption time is presented in (a) and decryption time in (b)

Table 3: Comparative analysis of EFDS with the existing schemes in terms of encryption time

File size in MB	EFDS	SDRF	MECC
5	4.37	6.04	9.62
10	8.72	9.48	10.59
15	11.51	13.38	16.44
20	13.29	15.34	17.46
25	15.81	17.29	21.9
30	17.51	19.22	23.35
35	19.14	20.16	27.29
40	21.63	22.63	28.49

5.2 Decryption Time

Fig. 3b elucidates the decryption time. In the case of a file size of 15 MB, the EFDS, SDRF, and MECC schemes take 10.56, 15.84, and 20.57 s for the decryption of all blocks in the file. The results show that the EFDS provides better file size encryption time than SDRF and MECC schemes. The comparison of the decryption time is presented in Table 4. It presents a comparative analysis of the scheme EFDS with other existing schemes for different sizes of data in MB. It can be observed from the table that for any data size, the scheme EFDS performs better than the existing scheme of SDRF and MECC. In the case of the data size of 30 MB, the decryption time is 16.81, 19.32, and 22.78 s for the schemes EFDS, SDRF, and MECC, respectively.

Table 4: Comparative analysis of EFDS in terms of decryption time

File size in MB	EFDS	SDRF	MECC
5	3.97	5.25	8.29
10	6.31	7.60	9.43
15	9.85	10.08	11.95
20	12.73	13.21	14.33
25	15.67	16.48	19.84
30	16.81	19.32	22.78
35	18.39	20.53	25.43
40	20.22	21.58	27.72

5.3 Block Generation Time

The proposed scheme is compared with existing schemes in terms of block generation time and data de-duplication time based on the size of the file. In Fig. 4a, the performance of EFDS is evaluated in terms of block generation time with different file sizes. In case the file size of 10 MB, the EFDS, SDRF, and MECC generate all blocks from the entire file in 3.58, 4.35, and 4.84 s, respectively. The comparison of the block generation time is presented in Table 5. The results prove that the EFDS utilizes less time for block generation in contrast to SDRF and MECC.

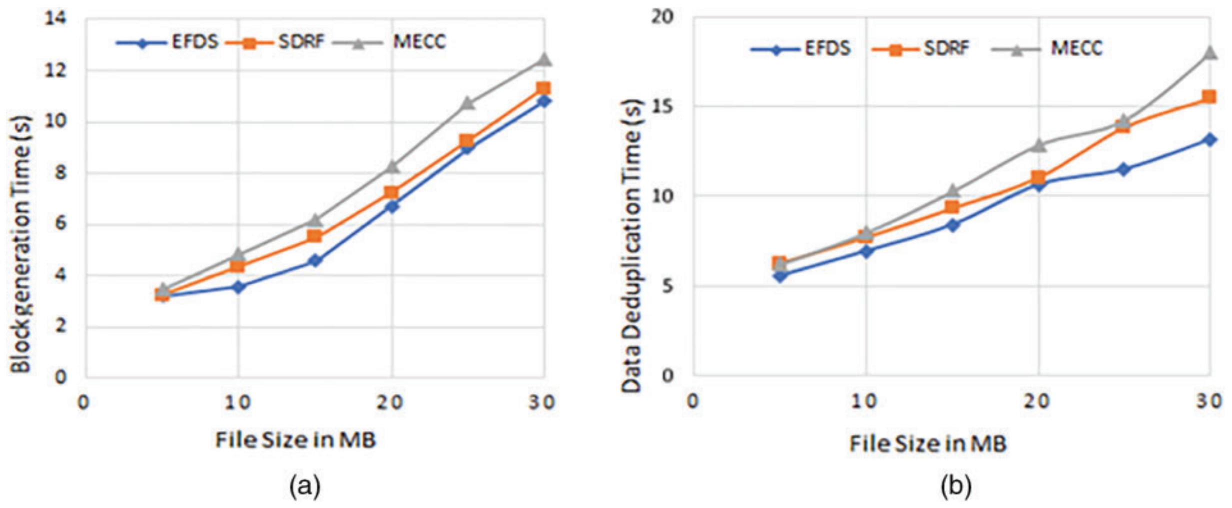

Figure 4: Block generation time is illustrated in (a) whereas (b) shows the data de-duplication time

Table 5: Comparative analysis of EFDS with the existing schemes for block generation time

File size in MB	EFDS	SDRF	MECC
5	3.21	3.23	3.76
10	3.58	4.35	4.84
15	4.58	5.49	6.20
20	6.73	7.24	8.25
25	8.97	9.28	1073
30	10.82	11.30	12.45

5.4 Data De-Duplication Time

In Fig. 4b, the performance of EFDS is evaluated in terms of data de-duplication time with different file sizes. In case the file size is 20 MB, the EFDS, SDRF, and MECC generate blocks for the entire file in 10.62, 11.05, and 12.85 s, respectively. The proposed scheme provides better data de-duplication time when compared with SDRF and MECC. The comparison of the data de-duplication time is presented in Table 6. In both of the figures, it has been observed that the de-duplication time and block generation time are increased when the size of the files increases.

Table 6: Comparative analysis of EFDS with the existing schemes for data de-duplication time

File size in MB	EFDS	SDRF	MECC
5	5.56	6.28	6.23
10	8.40	7.72	8.01
15	9.94	9.34	10.29
20	10.62	11.05	12.85
25	11.49	13.85	14.25
30	13.15	15.45	17.97

5.5 Block Operation Time

The proposed scheme is evaluated in terms of operation time and data de-duplication rate with different block sizes. In Fig. 5a, the operational time of the EFDS is evaluated against different block sizes and compares the performance with the SDRF and MECC schemes. In case the block size is 250 KB, the operational times are 135.6, 218.2, and 242.4 ms of EFDS, SDRF, and MECC schemes, respectively. The comparison of the block operation time is presented in Table 7. In the case of the data block size of 750 KB, the schemes EFDS, SDRF, and MECC take block operation times of 140.56, 203.49, and 239.28 ms, respectively. The results clearly show the better performance of the EFDS in contrast to its counterparts.

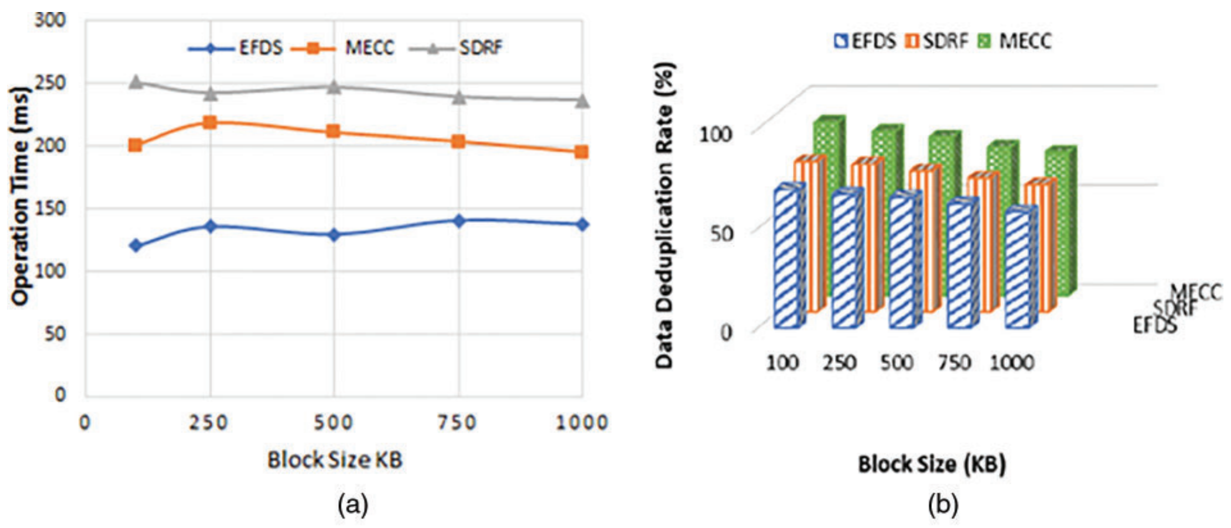

Figure 5: Block operation time in (a) and data de-duplication rate in (b)

Table 7: Comparison of EFDS with the existing schemes for block operation time

Data block size in KB	EFDS	SDRF	MECC
100	120.34	200.67	250.90
250	135.60	218.20	242.40
500	129.58	210.58	247.11
750	140.56	203.49	239.28

5.6 Data De-Duplication Rate

Fig. 5b elucidates the performance of EFDS is evaluated in terms of data de-duplication rate with various block sizes. In case the block size of 500 KB, the EFDS, SDRF, and MECC provide data de-duplication rates of 65.84%, 71.02%, and 79.97%, respectively. The results show that with the increase in the block sizes, the data rate decreases. However, the EFDS provides a better data rate when compared with SDRF and MECC schemes. The comparison of the data de-duplication rate is presented in Table 8. It can be observed from the table that of any data size in KB, the scheme EFDS performs is better than the existing scheme of SDRF and MECC. In the case where the data size is 500 KB, the data de-duplication rate is 62.63%, 67.34%, and 74.82% for EFDS, SDRF, and MECC, respectively. The EFDS scheme performance is better than the other existing scheme of the SDRF and MECC for the data de-duplication rate. The table explores that for all data sizes in KB, the EFDS scheme is Superior to the other existing scheme of SDRF and MECC.

5.7 Block Storage Time

In Fig. 6 the performance of the proposed scheme is evaluated in terms of block storage time based on the file size. In the case of a 15 MB file size, the data de-duplication time of EFDS, SDRF, and MECC is 23.43, 34.76, and 35.88 s, respectively. It can be observed that with an increase in file sizes, the required storage time also increases. Moreover, the EFDS provides better storage time compared with SDRF and MECC schemes.

Table 8: Comparative analysis in terms of data de-duplication rate

Data block size in KB	EFDS	SDRF	MECC
100	69.51	75.63	87.23
250	67.33	74.02	82.63
500	65.84	71.20	79.97
750	62.63	67.34	74.82
1000	58.49	63.99	72.20

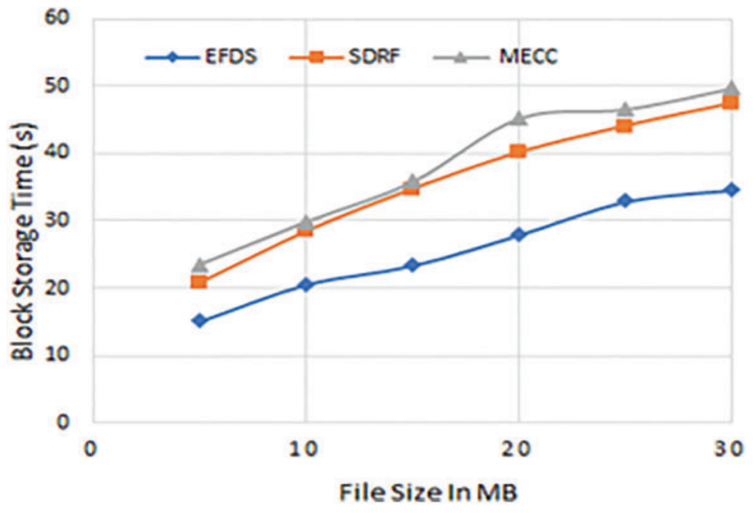

Figure 6: Block storage time

6 Conclusion

In this article, a strategy for duplicated data has been described. The duplicated data for SD-IoV is stored in a redundant form, which increases the communication cost and uses additional memory capacity. Data stored at the central repository should not be redundant and needs to eliminate unnecessary duplicates. In the existing schemes, tag generation is used for the data de-duplication file scheme based on Rabin fingerprinting for deterministic tags and random tags. By using MECC algorithms, the time it takes to generate a hash tree and tags for each file is extremely long which grows with file size. For the enhancement of the existing issues, the proposed scheme EFDS uses the novel HDBIS algorithm to reduce encryption, decryption time, block generation, data de-duplication time, block data rate, and block storage time. The experimental values show that the scheme EFDS performs better in contrast to its counterparts. Results show that EFDS achieves a de-duplication rate of 65.84% in contrast to 71.20% and 79.97% for SDRF and MECC for a data block size of 500 KB. In the case of encryption time, the EFDS achieves 11.51 s whereas the counterparts consume 13.38 and 17.46 s for a file size of 15 MB. The limitation of this work is that it focuses on a single file to identify duplicate blocks. In the future, researchers shall identify the duplicates in the already stored file in conjunction with the recent file.

Funding Statement: This work was supported in part by Hankuk University of Foreign Studies' Research Fund for 2023 and in part by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT Korea No. 2021R1F1A1045933.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. A. Ahanger and A. Aljumah, "Internet of things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2019.

[2] M. Kumar, N. Nayar, G. Mehta and A. Sharma, "Application of IoT in current pandemic of COVID-19," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, pp. 1–7, 2021.

[3] T. Hewa, G. Gur, A. Kalla, M. Ylianttila, A. Bracken *et al.,* "The role of blockchain in 6G: Challenges, opportunities and research directions," in *Proc. of 2nd 6G Wireless Summit 2020 Gain Edge 6G Era, 6G SUMMIT 2020*, Levi, Finland, pp. 1–5, 2020.

[4] G. Latif, J. M. Alghazo, R. Maheswar, P. Jayarajan and A. Sampathkumar, "Impact of IoT-based smart cities on human daily life," in *EAI/Springer Innovations in Communication and Computing*, Springer, pp. 103–114, 2020.

[5] M. Yousefpoor, E. Yousefpoor, H. Barati, A. Barati, A. Movaghar *et al.,* "Secure data aggregation methods and countermeasures against various attacks in wireless sensor networks: A comprehensive review," *Journal of Network and Computer Applications*, vol. 190, no. 2020, pp. 1–42, 2021.

[6] S. Loughney, J. Wang, D. Matellini and T. Nguyen, "Utilizing the evidential reasoning approach to determine a suitable wireless sensor network orientation for asset integrity monitoring of an offshore gas turbine driven generator," *Expert Systems with Applications*, vol. 185, pp. 1–15, 2021.

[7] E. GarciaCeja, C. Galván-Tejada and R. Brena, "Multi-view stacking for activity recognition with sound and accelerometer data," in *Proc. of the 9th ACM on Multimedia Systems Conf., MMSys'18, ACM*, New York, NY, USA, vol. 40, pp. 472–477, 2018.

[8] N. Luong, D. Hoang, P. Wang, D. Niyato, D. Kim *et al.,* "Data collection and wireless communication in Internet of Things (IoT) using economic analysis and pricing models: A survey," *IEEE Communications Survey and Tutorials*, vol. 18, no. 4, pp. 2546–2590, 2016.

[9] M. Mollah, J. Zhao, D. Niyato, Y. Guan, C. Uen *et al.,* "Blockchain for the internet of vehicles towards intelligent transportation systems: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4157–4185, 2021.

[10] S. M. A. Mohamed and Y. Wang, "A survey on novel classification of deduplication storage systems," *Distributed and Parallel Databases*, vol. 39, no. 1, pp. 201–230, 2021.

[11] U. Jayasankar, V. Thirumal and D. Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," *Journal of King Saud University—Computer and Information Sciencs*, vol. 33, no. 2, pp. 119–140, 2021.

[12] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari and H. Shakarami, "Data replication schemes in cloud computing: A survey," *Cluster Computing*, vol. 24, pp. 2545–2579, 2021.

[13] K. M. Vinoth, K. Venkatachalam, P. Prabu, A. Almutairi and M. Abouhawwash, "Secure biometric authentication with de-duplication on distributed cloud storage," *PeerJ Computer Science*, vol. 7, pp. 1–20, 2021.

[14] B. Latré, B. Braem, I. Moerman, C. Blondia and P. Demeester, "A survey on wireless body area networks," *Wireless Networks*, vol. 17, no. 1, pp. 1–18, 2011.

[15] M. H. ur Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah *et al.,* "Big data reduction methods: A survey," *Data Science and Engineering*, vol. 1, pp. 265–284, 2016.

[16] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi *et al.,* "Edge and fog computing for IoT: A survey on current research activities & future directions," *Computer Communications*, vol. 180, pp. 210–231, 2021.

[17] M. Heddes, I. Nunes, T. Givargis, A. Nicolau and A. Veidenbaum, "Hyperdimensional hashing: A robust and efficient dynamic hash table," *Association for Computing Machinery*, vol. 1, pp. 1–6, 2022.

[18] Z. Xia, X. Wang, X. Sun and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transaction on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[19] A. D. Yusuf, S. Abdullahi, M. M. Boukar and S. I. Yusuf, "Collision resolution techniques in hash table: A review," *International Journal of Advanced Computer Science and Application*, vol. 12, no. 9, pp. 757–762, 2021.

[20] Y. Zhang, H. Su, M. Yang, D. Zheng, F. Ren *et al.,* "Secure deduplication based on rabin fingerprinting over wireless sensing data in cloud computing," *Security and Communication Networks*, vol. 2018, pp. 1–13, 2018.

[21] W. Xia, D. Feng, H. Jiang, Y. Zhang, V. Chang *et al.,* "Accelerating content-defined-chunking based data deduplication by exploiting parallelism," *Future Generation Computer Systems*, vol. 98, pp. 406–418, 2019.

[22] R. N. S. Widodo, H. Lim and M. Atiquzzaman, "SDM: Smart deduplication for mobile cloud storage," *Future Generation Computer Systems*, vol. 70, pp. 64–73, 2017.

[23] Y. Zhang and C. L. P. Chen, "Secure heterogeneous data deduplication via fog-assisted mobile crowdsensing in 5G-enabled IIoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2849–2857, 2022.

[24] Y. Zhang, H. Jiang, D. Feng, W. Xia, M. Fu *et al.,* "AE: An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication," in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, Hong Kong, China, vol. 26, pp. 1337–1345, 2015.

[25] B. Rasina Begum and P. Chitra, "SEEDDUP: A three-tier secure data deduplication architecture-based storage and retrieval for cross-domains over cloud," *IETE Journal of Research*, vol. 26, pp. 1–18, 2021.

[26] Y. Gao, H. Xian and A. Yu, "Secure data deduplication for internet-of-things sensor networks based on threshold dynamic adjustment," *International Journal Distributed Sensor Networks*, vol. 16, no. 3, pp. 1–13, 2020.

[27] R. N. S. Widodo, H. Lim and M. Atiquzzaman, "A new content-defined chunking algorithm for data deduplication in cloud storage," *Future Generation Computer Systems*, vol. 71, pp. 145–156, 2017.

[28] J. Yan, X. Wang, Q. Gan, S. Li and D. Huang, "Secure and efficient big data deduplication in fog computing," *Soft Computing*, vol. 24, no. 8, pp. 5671–5682, 2020.

[29] L. Stephygraph and N. Arunkumar, "Brain-actuated wireless mobile robot control through an adaptive human-machine interface," in *Proc. of the Int. Conf. on Soft Computing System*, New Delhi, Springer, vol. 397, pp. 537–549, 2015.

[30] G. Lu, Y. Jin and D. H. C. Du, "Frequency based chunking for data de-duplication," in *Proc. of Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, Miami Beach, FL, USA, pp. 287–296, 2010.

[31] Y. Xuan Xing, N. Xiao, F. Liu, Z. Sun and W. Hui He, "AR-dedupe: An efficient deduplication approach for cluster deduplication system," *Journal of Shanghai Jiaotong University*, vol. 20, no. 1, pp. 76–81, 2015.

[32] V. Khanaa, A. Kumaravel and A. Rama, "Data deduplication on encrypted big data in cloud," *International Journal Engineering Advance Technology*, vol. 8, no. 4, pp. 644–648, 2019.

[33] N. Saleh, A. Kassem and A. M. Haidar, "Energy-efficient architecture for wireless sensor networks in healthcare applications," *IEEE Access*, vol. 6, pp. 6478–6496, 2018.

[34] P. G. Shynu, R. K. Nadesh, V. G. Menon, P. Venu, M. Abbasi *et al.,* "A secure data deduplication system for integrated cloud-edge networks," *Journal of Cloud Computing*, vol. 9, no. 61, pp. 1–12, 2020.

[35] A. S. M. Saeed and L. E. George, "Data deduplication system based on content-defined chunking using bytes pair frequency occurrence," *Symmetry*, vol. 12, no. 11, pp. 1–21, 2020.

[36] H. Li, J. Yu, H. Zhang, M. Yang and H. Wang, "Privacy-preserving and distributed algorithms for modular exponentiation in IoT with edge computing assistance," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8769–8779, 2020.

[37] S. Li, T. Lan, B. Balasubramanian, H. W. Li, M. R. Ra *et al.,* "Pushing collaborative data deduplication to the network edge: An optimization framework and system design," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 1–11, 2022.
[38] S. E. Ebinazer, N. Savarimuthu and S. Mary Saira Bhanu, "An efficient secure data deduplication method using radix trie with bloom filter (SDD-RT-BF) in cloud environment," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 2443–2451, 2020.
[39] Y. Yang, S. R. Kuppannagari and V. K. Prasanna, "A high throughput parallel hash table accelerator on HBM-enabled FPGAs," in *Proc. of Int. Conf. on Field-Programmable Technol. ICFPT*, Maui, HI, USA, pp. 148–153, 2020.
[40] D. Tong, S. Zhou and V. K. Prasanna, "High-throughput online hash table on FPGA," in *Proc. of 2015 IEEE Int. Parallel and Distributed Processing Symp. Workshops*, Hyderabad, India, pp. 105–112, 2015.
[41] K. Halba and C. Mahmoudi, "In-vehicle software defined networking: An enabler for data interoperability," in *Proc. of the 2nd Int. Conf. on Information System and Data Mining*, Lakeland, FL, USA, pp. 93–97, 2018.
[42] M. Haeberle, F. Heimgaertner, H. Lockr, N. Nayak, D. Grewe *et al.,* "Softwarization of automotive e/e architectures: A software-defined networking approach," in *Proc. of the 2020 IEEE Vehicular Networking Conf. (VNC)*, New York, NY, USA, vol. 2020, pp. 1–8, 2020.
[43] W. Chen, Z. Chen, D. Li, H. Liu and Y. Tang, "Low-overhead inline deduplication for persistent memory," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, pp. 1–13, 2021.