IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 2, 2021, accepted January 7, 2022, date of publication January 18, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144632

Towards Big Data Security Framework by Leveraging Fragmentation and Blockchain Technology

HANAN E. ALHAZMI[1,2], FATHY E. EASSA[1], AND SUHELAH M. SANDOKJI[1]
[1]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah 21589, Saudi Arabia
[2]Computer Science Department, Umm Al-Qura University, Makkah 21955, Saudi Arabia

Corresponding author: Hanan E. Alhazmi (hehazmi@uqu.edu.sa)

ABSTRACT The world is facing a growth in the amount and variety of data generated by both users and machines. Despite the exponential increases, the tools and technologies developed to manage these data volumes are not intended to meet security and data protection requirements. Additionally, most of the current big data security systems are offered by a centralized third party, which is vulnerable to many security threats. Blockchain technology plays a significant role by addressing modern technology concerns such as decentralization, non-tampering, trust, data ownership, and traceability, making it great potential to protect personal information. This research presents a new big data security solution empowered by blockchain technology and incorporates fragmentation, encryption, and access control techniques. Our proposed fragmentation algorithm takes into account the data owner's demand for encryption to be added to the fragmentation process. Furthermore, data fragments will be stored in the distributed manner offered by the big data environment, resulting in an additional layer of data protection. In order to achieve an optimal security solution, we aim to enhance big data security with acceptable overhead and avoid the encryption overhead for non-sensitive and low-sensitive data portions. We present the results of our implemented techniques to highlight that the overheads (in terms of computation time) introduced by our solution are negligible relative to its security and privacy gains.

INDEX TERMS Big data security, blockchain, fragmentation, access control, auditing.

I. INTRODUCTION

The volume of data in our world is increasing rapidly. According to International Data Corporation (IDC) prediction [1], global data generation is expected to reach more than 180 zettabytes by 2025. In the big data environment, data is regularly being collected and analyzed. Companies and organizations usually use the collected data to personalize services, improve decision-making optimization, predict future trends, etc. Nowadays, data is an essential factor in the industry [2]. When big data is stored, the data is usually stored in a distributed file system or the cloud. In distributed storage, multiple nodes cooperation is needed to accomplish a specific task. Thus, attacking one or several nodes will affect the reliability of computing results. Distributed data storage dramatically increases the burden of protection on storage nodes. If data encryption storage is applied, key management will be more complicated. Accordingly, in big data schemes, it is hard to apply the traditional asymmetric, and symmetric encryption techniques directly [3]. In the case of cloud storage, due to large scale volume, significant increment, and rapid changes, encrypting big datasets directly will raise the risk of secret key management and require a considerable computation overhead [4]. Once the secret key is exposed, the whole dataset can be corrupted and stolen. Blockchain technology has attracted academics and industry due to several properties such as immutability, confidentiality, integrity, and authorization [5]. It is based on public and private key infrastructure (PKI), cryptography, peer-to-peer network, and consensus algorithms that keep the ledger's immutability.

Combining blockchain with big data brings many benefits. For instance, the automation provided by blockchain workflow will help companies become more efficient and productive [6], [7]. The two technologies are significant to the industry, and the blockchain is still new to be used in data

The associate editor coordinating the review of this manuscript and approving it for publication was Lo'ai Tawalbeh.

science; therefore, more efforts are required. Any effective security system should meet the main security properties [8], which are confidentiality (preventing unauthorized access and infringement), availability (assuring access to resources by authorized users in both normal and disastrous situations), and integrity (avoiding resources to be modified without authorization). Information protection can be carried out in several ways: by 1) strict access control techniques (using authentication or access control policies), 2) hiding information (using encryption, steganography, and scrambling), and 3) making it understandable and available only by people who own a map or key to reverse transformation efficiently and return the original version.

Data fragmentation by means of data security is not new and has been used in numerous studies [9], [10]. Moreover, fragmentation is combined with encryption [11], [12] to enable parallel encryption of information pieces, which is more efficient than executing complete sequential encryption. More recent studies suggested using fragmentation to eliminate encryption for data protection in order to avoid significant overhead [13], [14]. However, Any data security strategy should include encryption as a cornerstone [15]. Consequently, our solution recommends encryption for data at rest, especially for high-sensitive data. The contributions of this paper can be summarized as follows:

- This paper's main contribution is to propose a new blockchain-based framework architecture for securing a big data system.
- we propose new fragmentation techniques which facilitate the encryption regarding user preferences.
- We designed a lightweight metadata structure to be stored in the blockchain ledger.
- We propose a new access control mechanism based on blockchain.
- We propose a blockchain-based big data auditing method that eliminates the third-party auditing, hence improving the dependability and stability of auditing methods.
- We show that the security, efficiency, and reliability of our solution are favorable.

The rest of this paper is organized as follows: The background to big data and blockchain is briefly discussed in Section II. In addition, Section III presents the related work. Then, our proposed framework is discussed in Section IV. The results and experiments are provided in Section V. After that, Section VI presents framework evaluation, and, finally, Section VII summarizes the paper.

II. BACKGROUND

A. BIG DATA TECHNOLOGY

Generally, big data can be defined as the answer to the question as to how difficult it is to handle big datasets, with different forms, including structured, semi-structured, and unstructured, which are difficult to handle using traditional data analysis techniques [3], [16]. The other definition comes from ''what big data is,'' which focuses on a data processing technology architecture to capture and analyze valuable information from large and varied datasets [3]. These two definitions represent the five dimensions of big data (Volume, Variety, Velocity, Veracity, and Value) [17], [18], also referred to as the ''5Vs'', which are described as follows:

1) VOLUME

Volume refers to the extreme quantity of data. Many issues arise when processing big data, such as the difficulty of storing or loading the whole dataset in memory and hard drive, as well as the problem of dimensionality (multiple features and attributes).

2) VARIETY

There are several different types of data, including video, audio, and text, classified as structured, semi-structured, or unstructured data. Data heterogeneity, data locality, noisy and dirty data are all issues that come with variety.

3) VELOCITY

Velocity measures the flow and speed of data. In the near future, many portable devices will rise to 13.1 billion in 2023, which means a massive amount of traffic. The vast traffic also comes from the growth of high-definition videos, video streaming, and video gaming.

4) VERACITY

Veracity is the degree of data quality. In other words, data must be accurate, precise, and trusted. More research efforts must be dedicated to data provenance, dirty and noisy data elimination, and data uncertainty to improve the quality and accuracy of big data analytic.

5) VALUE

In the world of big data, value is an essential aspect. Big data is rarely useful on its own; thus, to make it valuable, it must be transformed into knowledge, where data processing comes in.

B. BIG DATA SECURITY ISSUES

Big data reflects big problems in security as well as management. Its characteristics brought many challenges, especially with the rise of IoT devices that produce a massive amount of data, which needs processing to introduce valuable information and new services such as on-demand computing power and storage from public cloud providers. Hence, big data security becomes an essential need of any big data system. Furthermore, the security techniques currently in use, for example, firewalls and other perimeter security tools, are not effective in a big data infrastructure. In this paper, we summarize different big data security issues as follows:

- Preventing attackers from generating and adding fake data.
- Access control mechanisms such as granular access control (grants different users with different access levels) are no longer useful with big data.

• Distributing the storing and processing of big data among multiple machines brought many security risks. For example, identifying an attack could require a long time to detect.

• Security auditing is a critical technical assessment for any organization. However, it is rare to be implemented in big data systems. As a result, it becomes challenging to manage data provenance, which is helpful to detect where breaches come from.

C. BLOCKCHAIN TECHNOLOGY

Securing and processing large-scale data is certainly not a simple task. Blockchain could be considered as an ideal solution for addressing many challenges of big data analytics and management. Blockchain provides essential properties such as decentralization, integrity, and immutability.

• **Blockchain**: A blockchain [19] is a distributed ledger of transactions or events recorded and stored in blocks that are linked using cryptography. The records are shared and monitored by all network nodes, updated only by miners, which are nodes with powerful computational resources that validate new transactions and record them on the ledger. The blockchain is made up of timestamped blocks holding transactions in which each block is linked to a previous block.

• **Blockchain Network**: The underlying infrastructure of blockchain is a peer-to-peer network. Every peer in the network follows certain peering protocol rules, transaction processing, consensus protocol, and ledger management. The blockchain has three types: public, private, and consortium blockchains. The blockchain type can be differentiated based on the authorization level of a node in a network in terms of being a validator and accessing the blockchain data. A permissionless blockchain is public and allows anonymous users to cooperate with their resources, such as bitcoin and Ethereum. Private blockchains are centralized and open to individuals or entity while consortium blockchains are partially decentralized and open to specific organizations. Private and consortium blockchains are categorized as permissioned blockchains.

• **Data Structure of Blockchain**: A block is a data file that records any type of transaction on the network. Figure 1 shows the structure of the blockchain. Once data is stored in a block and becomes a part of the chain, tampering is impossible. Each block holds a group of valid and anti-double spending transactions. The transactions are constructed as Merkle tree. In this manner, each block includes one Merkle tree. Additionally, the Merkle tree is located in the block header. The block header consists of several elements, such as the hash of the previous block and timestamp.

• **Blockchain Transaction**: For instance, a bitcoin transaction transfers token ownership from the sender to the receiver account. It can be described as a public static data record that represents a token value transfer between two peers. It defines set of inputs and outputs. Each input holds a previous UXTO (unspent transaction output) related to the sender and his signature to the inputs.

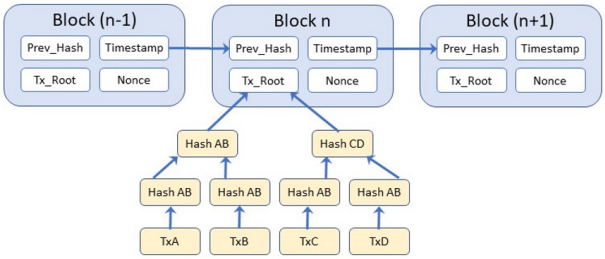

FIGURE 1. Blockchain structure.

• **Blockchain Consensus Algorithms** : Consensus mechanisms/algorithms are used to ensure the agreement among blockchain nodes/peers. All participant peers must agree on the serialized transaction history stored in the blockchain ledger. Different classes of blockchain use separated categories of consensus algorithms. For instance, permissionless blockchain uses various consensus algorithms based on a challenge-response strategy such as proof of work and proof of stake [20]. On the other hand, permissioned blockchain implements variations of the Byzantian Fault Tolerance (BFT) algorithm [21]. For example, Practical BFT (PBFT) [22] and Redundant BFT (RBFT) [23].

Generally, the blockchain consensus algorithm includes the following steps [24]:

• Block proposal which involves generation and attaching cryptographic proofs.

• Block and transaction broadcasting/ advertising across the network.

• Block validation by checking the transaction validity and the generation proofs.

• Block finalization by reaching an agreement on the validated block acceptance.

• Incentive technique to promote the honest participants and produces network tokens rewards.

III. RELATED WORK

Many studies discussed blockchain technology with medical data access in healthcare, which calls that people must own and access their health records. Blockchain has the potential to provide secure electronic health record (EHR) sharing in which the patients are the actual owners. In [25], authors proposed that the blockchain stores only the metadata related to medical and health events. Thus, avoiding enormously scaled blockchain infrastructure to store the entire health records.

Human activities in social media networks produce large-scale databases. There is a high risk of relying on third parties to protect personal and sensitive data. Hence, there is a need to let users track and control their online activities. Blockchain can be used as a permission-filtering technique in social networks as proposed Ushare [20]. Ushare is a social media framework based on blockchain and provides user privileges to control his data access. PCA (personal certificate authority) is created for each post to manage the user's circles and encrypt data before broadcast. Blockchain is used to store the

TABLE 1. A Summary of Blockchain-based solutions for big data.

Ref.	Description	Focus	Domain	Blockchain Purpose
[25]	Blockchain solution to enable an exchanging of electronic health records in which health consumers are the ultimate owners.	Medical data access and sharing	Big data in health-care	Storing metadata related to medical events
[20]	Ushare: user-controlled social media network based on blockchain in which users have privileges to control the access and share of their data	Data sharing Access control	Social media	Storing the transactions related to users' shares.
[26]	Blockchain-based framework for adapting to the limited resources in edge devices a consensus algorithm: Proof-of-Collaboration (PoC)	Big data sharing	Edge computing	Solving distrust issues of big data sharing in collaborative edges.
[27]	Blockchain-based access control framework	big data access control	General big data	Using Smart contract to code access control policies.
[28]	Decentralized big data auditing scheme	Big data auditing	Smart cities	Enhancing the stability and reliability of auditing schemes by excluding the need for centralized third-party auditing.
[29]	Auditing scheme for ensuring the integrity of data in cloud storage	Big data auditing	Cloud storage	- Eliminating the centralized third-party auditor. - Storing the tags of data integrity verification on the Markel tree to reduce communication and computation overhead.
[30]	Blockchain Based Big Data Security Protection Scheme	Big data access control	General big data	Enhancing Hadoop security by: Improve heartbeat model Maintain the security of metadata.

transactions related to users' shares. In spite that Ushare as a user-centric blockchain application could enable end-users to control their data, their solution is difficult to implement in terms of producing a huge number of transactions stored in the blockchain as well as encrypting the entire content of user's data.

The work in [16] aims to introduce a blockchain framework for smart mobility transactions. The purpose is to secure and protect the collected data of individuals for different participants such as companies, governments, and universities. In this scenario, each participant shares their encrypted data to a blockchain network issued with a smart contract containing transaction rules. This framework is limited to smart mobility scenarios.

Z. Guan *et al.* [31] proposed a big data collection and trading system based on the Blockchain and Trusted Security Model (TSM). They combined different technologies such as Physical Unclonable Function (PUF), which uses the fingerprint as a sensor identifier while the TSM model ensures the data collection process's trust. They aim to allow trading private data and moderate any attack by using blockchain to facilitate accounting and trading processes. Trust is an essential issue in edge computing among edge devices that share big data. However, their solution lacks considering the end-to-end data protection.

In another paper by C. Xu *et al.* [26], a blockchain-based framework in collaborative edges is proposed to allow trusted big data sharing while maintaining efficient resource usage in edge devices. Moreover, this work proposed a consensus algorithm called Proof-of-Collaboration (PoC) to achieve optimized computational power. Smart environments produce a large amount of data, typically are personal and sensitive data that require great attention to secure and protect.

To avoid centralization in big data auditing, which the third party regularly provides, the study in [28] proposed a scheme for decentralized big data auditing in smart cities based on blockchain to improve stability and reliability to participate in smart city construction.

Another paper by S. Li [32] integrates big data, the energy Internet, IoT, and blockchain to construct a smart city. They exploit blockchain features that are compatible with the nature of the energy Internet. Therefore, the issue of expensive maintenance of centralized databases in big data centers is solved. A smart contract is a self-executed computer code stored in the blockchain and executed based on certain conditions evaluation. Smart contracts in blockchain technology offer a new solution to trust issues in big data by automatically executing default instructions. Even though the work offers redundant and distributed storage, it suffers from high cost, poor recovery capability, high cost of maintenance of IoT equipment.

In [33], the authors presented a big blockchain-based data-sharing framework and exploited smart contracts to secure big data sharing.

Access control and authentication are considered key technologies to solve the privacy and security problems in big data. Es-Samaali *et al.* [27] use blockchain technology to develop a big data access control solution. They use smart contracts to code access control policies to check authorization for big data access requests. They adopt blockchain to enforce access policies in distributed environments where there is no central authority.

In [28], the authors proposed decentralized big data auditing scheme based on blockchain for smart cities. Their goal is to enhance auditing schemes' stability and reliability by excluding the need for centralized third-party auditing but the overhead to the users is significant. Their method produces extra costs in blockchain ledger navigation during the auditing process. Similar work has been presented in [29]; however, they designed their scheme for cloud storage. Moreover, they proposed to store the tags of data integrity verification on blockchain to reduce the overhead of communication and computation produced by the integrity verification process.

Authors in [30] proposed a big data access control scheme to enhance Hadoop security by maintaining metadata security and improving the heartbeat model.

Table 1. summarizes various blockchain-based solutions for big data.

Generally, exploiting blockchain technology in the era of big data security and management is new and requires more effort. Even though previous studies employed blockchain for data sharing and access control solutions, it was restricted to specific domains and applications in big data. Furthermore, the proposed solution must also be integrated with access control, data security at rest, in transit, and auditing to improve big data security. However, this integration is limited in prior research. This study aims to address these limitations by proposing a general and comprehensive blockchain-based solution for managing and securing big data.

IV. PROPOSED FRAMEWORK

The proposed framework will be discussed in this section. Firstly, the system components will be introduced. Next, the workflow will be described. Finally, we will present our proposed techniques.

A. FRAMEWORK ARCHITECTURE

In our framework, data will be analyzed to define sensitive parts. According to user preferences, the sensitivity level will be determined. Depending on the sensitivity level, data will be treated differently. Data will be fragmented and encrypted before insertion into big data distributed storage. The metadata (MD) generated during the fragmentation process and the permission list (PL) will be held on the permissioned blockchain to facilitate search and tamper-resistant capabilities. The structure of MD and PL are shown in Tables 2 and 3.

Figure 2 illustrates the overall architecture of the proposed framework. The roles and responsibilities of entities in the proposed framework are described as follows:

- **The Data Owner (DO)** refers to the entity that owns the data and seeks to access or store it. DO has complete authority over his/her data. DO must create an access control policy for his/her data, including authorization for others to access his/her data.
- **The User (U)** is the entity requesting data access with granted authorization.
- **The BlockChain based Security Manager (BCSM)** ensures the authenticity of all events performed within the system. The events involve storing big data, storing metadata, and accessing the assets and logs on the ledger. The BCSM is also in charge of blockchain management. BCSM and other entities will communicate through a secure SSL/TLS channel.
- **Big data Distributed Storage (BDS)** After fragmentation and encryption, BDS is in charge of storing the big data.
- **The blockchain (BC)** is in charge of storing MD and the PL in the ledger. Moreover, BS is responsible for recording the access log and the other events of the system.

The proposed framework consists of the following key components:

1) DATA SENSITIVITY DETECTOR(DSD)

Sensitivity detection techniques can be categorized as automated, semi-automated, and manual. Due to the large volume of data, manual data sensitivity detection needs intensive efforts. Therefore, automated solutions are recommended. These solutions involve complex techniques that are out of this paper's scope and remain as future work. Examples include domain experts and neural networks such as in [34]–[36]. In our case, sensitivity detection is subjective to the data owner (DO) policy and requirements. The DO determines the sensitivity level of data (high, low, and not) and identifies the sensitive attributes to be protected. The flow of DSD is shown in Figure 3.

2) DATA SPLITTER(DS)

Our solution exploits the fragmentation techniques for providing an additional layer of securing sensitive data. Data is split up into sensitive and non-sensitive collections based on user requirements. The checksum is used to ensure the data integrity by calculating the SHA-512 [37] for the original file then comparing the result of hashing to the result of the file after the reconstruction process. Our framework handles the security of sensitive data according to the sensitivity level. For low-sensitive data, we use scrambling to harden the fragmentation process, which is combined with a distributed big data storage partitioning [14]. Additionally, our solution encrypts only the high-sensitive portion of the dataset to avoid the significant overhead associated with encrypting the whole volume of data. Further details of the proposed fragmentation techniques are presented in subsection B.

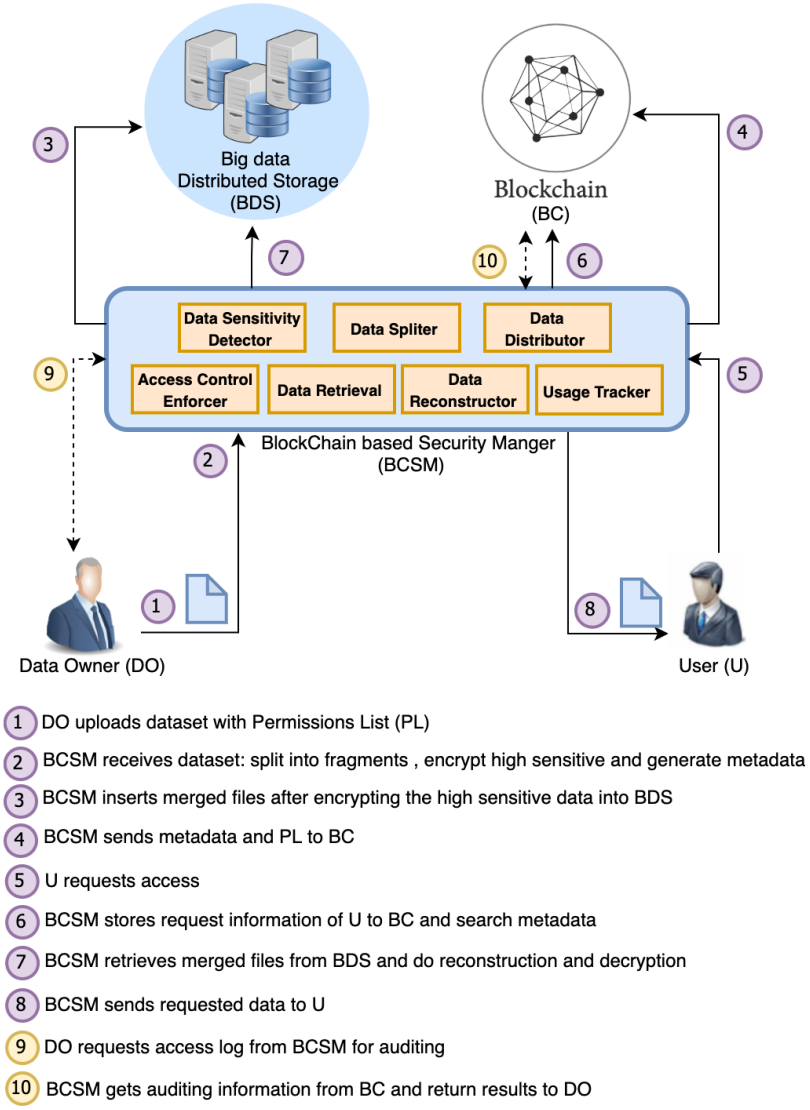

FIGURE 2. The proposed framework architecture.

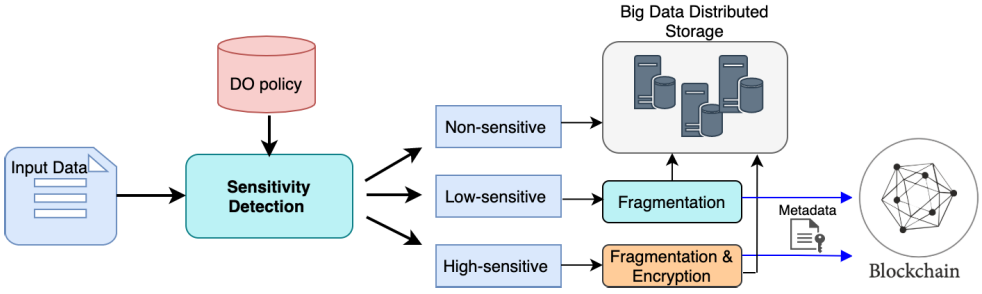

FIGURE 3. The flow of data sensitivity detector.

3) DATA DISTRIBUTOR(DD)

This component assigns the data-id and maps it to the merged files. Then DD creates and encrypts the MD. The size of the MD data structure depends on the number of fragments. Furthermore, this component sends the merged files to BDS. Also, it sends MD and PL to be stored on the blockchain ledger.

4) DATA RETRIEVAL(DR)

This component retrieves the information (metadata and data-hash) related to the data from the blockchain using the data-id, and the requested merged files are retrieved from the BDS. The DR then performs the decryption process of metadata in order to send them to the Data Reconstructor.

TABLE 2. Metadata structure.

MD Structure
dataset-id
Flag if high sensitive
Data-Hash
First fragment ID
Second fragment ID
.....
Last fragment ID

TABLE 3. Permission list (PL).

PL Structure
dataset-id
Owner name
Group Name
Owner Permissions
Group Permissions
other Permissions

5) DATA RECONSTRUCTOR(DRE)

This component returns the data to its original form using the metadata retrieved from the blockchain. It performs the decryption and defragmentation techniques to reconstruct the original file.

6) ACCESS CONTROL ENFORCER(ACE)

This component is responsible for the authentication and authorization of the data owner and user. The client applies for authentication to the ACE, and once authentication is granted, the authorization process is initiated. ACE checks the user authenticity by using multi-factor authentication. The ACL rules are used to enforce that the data is only accessed with privileges listed in PL. The requested data can only be accessible by a group of authorized users based on the PL. If the user is granted, the ACE records the request in the blockchain for auditing purposes.

7) USAGE TRACKER(UT)

This component retrieves auditing information from blockchain regarding data access and usage upon data owner or auditor request leveraging the traceability feature provided by the blockchain.

The interactions among these components are illustrated in Figure 4, which shows active components during uploading operation. Figure 5 shows active components during the reading operation.

The procedure of uploading and retrieving the data will be described in the following.

- **Uploading data**: The data storing operation begins with identifying the sensitivity of the data, as depicted in Figure 4. Next, the data is identified as high/low-sensitive and non-sensitive. Non-sensitive data will be stored in BDS, whereas sensitive data will be passed to the data splitter, which will split data into fragments and create the merged files. In the case of high-sensitive data, data fragments will be encrypted before creating the merged files. Moreover, the MD and PL

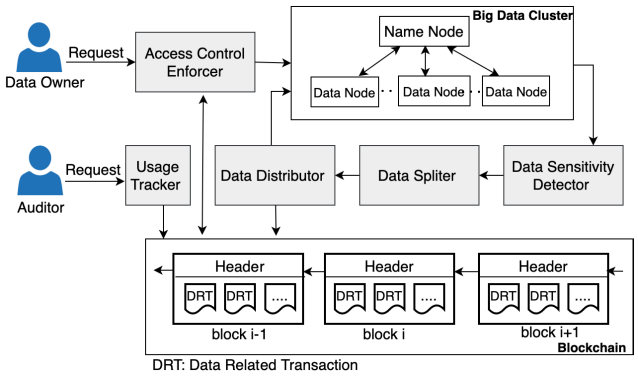

FIGURE 4. The active components of the framework during uploading.

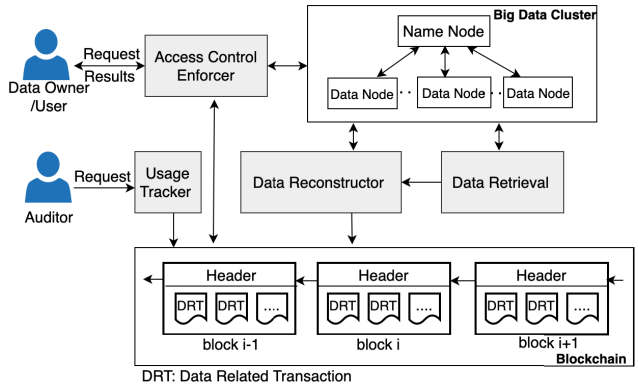

FIGURE 5. The active components of the framework during reading.

will be created to be stored and managed by the blockchain. Finally, the merged files will be partitioned based on block size then stored in BDS, which adds a layer of protection. Each transaction related to data such as access, read and write events will be recorded in the blockchain ledger as DRT (Data Related Transaction). This process is shown in Figure 6.

- **Retrieving Data**: In order to retrieve data, the user must be authenticated and authorized by checking the PL file associated with the requested data from the blockchain. After authorization is granted, the manager obtains the information of the requested data using the metadata retrieved from the blockchain, as shown in Figure 5. After determining the data's sensitivity, the manager reconstructs the data into its original form by defragmentation and decryption if needed. This process is illustrated in Figure 7.

B. PROPOSED FRAGMENTATION TECHNIQUES

Our fragmentation techniques aim to increase the security for two types of sensitive data, namely low or high-sensitive data. Based on the user's preference for handling his/her data, the degree of sensitivity is determined. This section presents our proposed fragmentation techniques which are fragmentation, defragmentation, fragmentation & encryption, and defragmentation & decryption. The first two techniques are applied for low-sensitive data and the other for high-sensitive data.

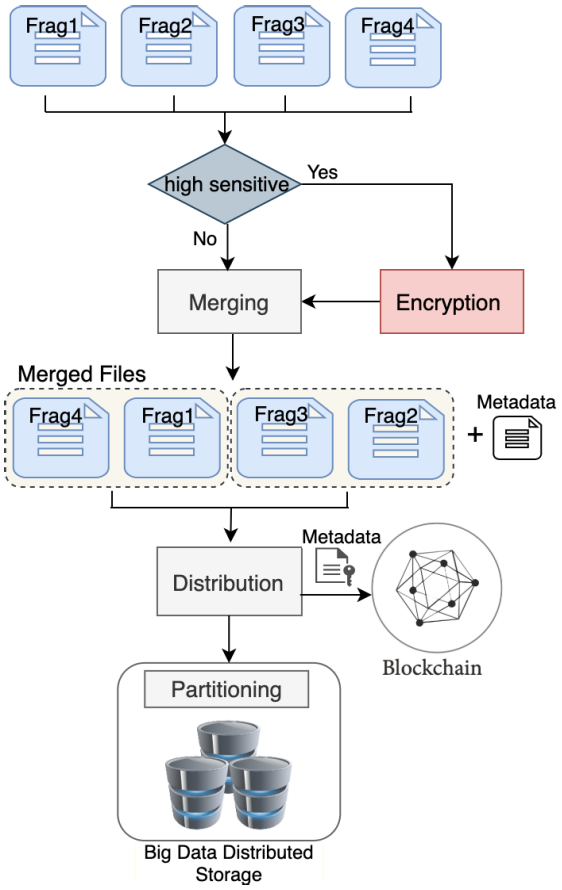

FIGURE 6. Data Upload process.

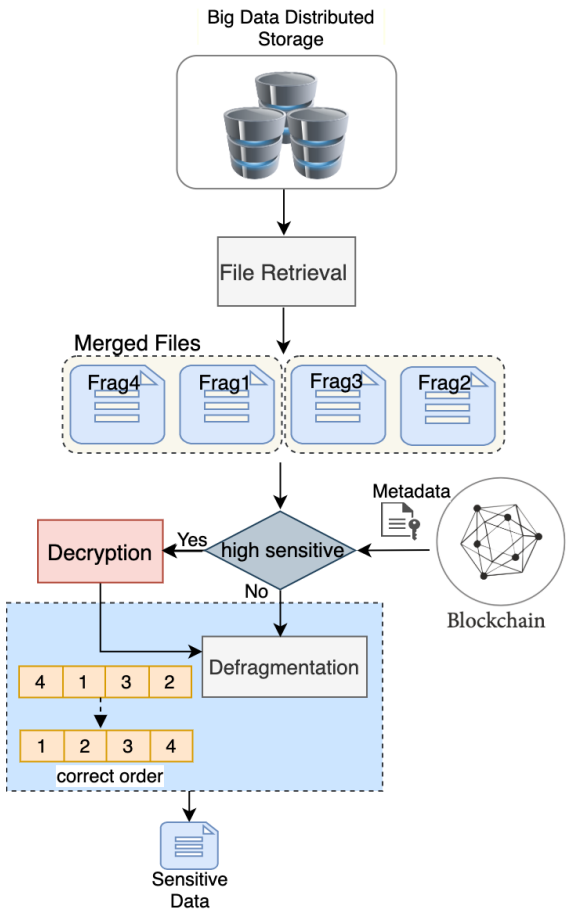

FIGURE 7. Data retrieve process.

1) FRAGMENTATION

Fragmentation tolerates efficient scrambling and encryption mechanisms utilizing parallelism. It is indispensable to consider the size of fragments; a too-large fragment may include too much information to reveal, whereas a too small fragment may cause agonizing overhead. To achieve optimal fragmentation, fragmentation must abide by the following rules:

- **Completeness**: Data should not be lost during the fragmentation process. Every data item must exist in at least one fragment.
- **Reconstruction**: If the data breaks down, the data can still be combined without any modification to the data structure. For all n fragments: $f_1 \cup f_2 \cup \ldots f_n = Originalfile$.
- **Disjointness**: Data in the fragment should not be included in the other fragments to prevent data duplication.

For each two fragments: f_i and f_j, where $i \neq j, f_i \cap f_j = \phi$.

The checksum is applied to ensure meeting the above rules by calculating the SHA-512 for the original file then comparing the result of hashing to the result of the file after the reconstruction process. Our framework handles the security of sensitive data according to the sensitivity level. For low-sensitive data, we use scrambling to harden the fragmentation process, which is combined with a distributed big data storage partitioning [14].

Our technique encrypts only the high-sensitive part of a dataset to avoid the high overhead produced by encrypting large-scale data. In any scenario, fragmentation for security raises latency inside the system. As a result, to achieve acceptable overall performance, a good fragmentation technique must be combined with processing parallelization. The fragmentation stores metadata of the dataset and its fragments. We propose a new data structure of metadata to be stored in blockchain ledger utilizing blockchain immutability and tamper resistance in order to assist the data integrity checks. The data-hash is calculated via the hashing algorithm (SHA-512) as: $message\ digest(md) = H(m)$.

The message digest (md) is calculated and added to the MD.

The pseudo-code for the fragmentation is illustrated in Algorithm 1. This algorithm has two major procedures. Firstly, splitting the sensitive data file into fragments based on predefined size suitable for the original file size. The original order of the fragments will be kept in the *MappingArray* by storing the ID of each fragment.

Creating the fragments will be based on the following:

- $\forall f_i, f_j \in F, \exists\ MappingArray[i] \wedge MappingArray[j]$
- $\forall f_i \neq f_j, MappingArray[i] \neq MappingArray[j]$,

where F is the fragments set and f_i, f_j are any two elements in the F set.

Algorithm 1 Fragmentation

Input *SenstvDataFile*, *fragmentSize*, *FilePath*, *IsHighSensitive*
Output *M:{* $m_1, m_2, m_3, \ldots m_n$*}*, *MappingArray* [ ]

procedure Fragmentation
  SenstvDataFile ← *FileInputStream*(*FilePath*)
  $F \leftarrow$ *SenstvDataFile.SplitFile*(*fragmentSize*)
  for $i \leftarrow 0$, *fragmentsNumber* **do**
    MappingArray[i] = f_i*.id*
  if *IsHighSensitive* **then**
    for $i \leftarrow 0$, *fragmentsNumber*/2 **do**
      m_i = *SenstvDataFile.merg*(*en*(f_j), *en*(f_k))
        ▷ f_j, f_k *are selected arbitrarily*
    end for
    mergeFileNumber = i
  else
    for $i \leftarrow 0$, *fragmentsNumber*/2 **do**
      m_i = *SenstvDataFile.merg*(f_j, f_k)
        ▷ f_j, f_k *are selected arbitrarily*
    end for
    mergeFileNumber = i
  end if
procedure Store
  for $i \leftarrow 0$, *mergeFileNumber* **do**
    insertHDFS(m_i)
  end for
  send MappingArray [] to blockchain

Then, the algorithm creates the merged files which consists of two fragments f_j and f_k as follows:

- f_j and f_k are selected arbitrarily.
- $\forall\ m_i \ni M, \exists f_j \cup f_k = m_i$ *and* $f_j \cap f_k = \phi$,

where M is the merged files set and m_i, m_j are any two elements in the M set.

The high-sensitive fragments will use AES encryption [38]. Finally, each merged file m will be inserted in HDFS.

2) DEFRAGMENTATION

Defragmentation is an inverse process of fragmentation. It follows the same rules of fragmentation. Algorithm 2 de- scribes our pseudo-code of defragmentation. The algorithm starts by getting the *MappingArray* embedded in MD from blockchain. Next step is searching for the *fID* in M to reconstruct the original file follows the original order found in *MappingArray*. For high-sensitive fragments, decryption is needed to form the original version.

V. EXPERIMENTS AND RESULTS

In this section, the experiments and results will be discussed in detail.

Algorithm 2 Defragmentation

Input *M:{* $m_1, m_2, m_3, \ldots m_n$*}*, *IsHighSensitive*, *MappingArray* [ ]
Output *SensitiveDataFile*

procedure Defragmentation
  get MappingArray[ ] from blockchain
  Outputstream=FileOutputStream(SensitiveDataFile)
  for $i \leftarrow$ *fragmentsNumber*, 0 **do**
    fID = *MappingArray*[i]
    Search fID in M if match, read fragment and send to Outputstream
    f = *FileInputStream*(*fID*)
    f*.read*()
    if *IsHighSensitive* **then**
      outputstrem.write(*decrypt*(f))
    else
      outputstrem.write(f)
  end for
  outputstream.close()

A. ENVIRONMENT SETUP

To evaluate the proposed techniques in the BMBD framework, we used a virtual Hadoop cluster consisting of one Name node and three data nodes. We also ran the experiments in a well-controlled virtual network to eliminate the factor of varying network conditions. A physical node is represented by a virtual machine, and all virtual machines run in the same host machine. The host machine is equipped with Intel Core i9 2.3 GHz, 16GB DDR6 memory, and a 1TB SSD hard disk. The virtual machine manager is VirtualBox 6.1.26. The Name node's hardware configuration is 8VCPU and 11.5 GB of RAM. For data nodes, we used 1VCPU and 1024MB. Each guest machine runs Ubuntu 18.04.5 LTS and Hadoop 3.3.0. We compared the overhead imposed by our proposed framework to another Hadoop cluster with the same hardware configuration but without the proposed framework. The replication factor in both Hadoop clusters is 3, and the data block size is 64MB.

B. DATA USED

We used structured datasets to evaluate the performance of the proposed framework. We used Synthea [39], which is a synthetic patient data generator. We have used different sizes of data files (64MB, 128 MB, 256 MB, 512 MB, and 1024 MB). We have implemented a prototype of our framework in Java to interact with the Hadoop cluster.

C. RESULTS

The experiments were conducted to evaluate the performance overhead caused by introducing our framework into HDFS. We measured the writing and reading speeds of the HDFS with and without the proposed techniques.

1) PROFILE OF EXPERIMENTS

We have developed a prototype of our framework in Java that is able to read and write to HDFS. We considered different structured files sizes to evaluate the performance of our framework. We have simulated the parallelism of fragmentation by implementing multithreading. Each experiment was repeated five times, and we took the average as the measured result.

Writing and reading experiments. We measured the computation time of copying a file from the local file system to HDFS and computed the speed in megabytes per millisecond. Moreover, we estimated the computation time of copying a file from HDFS to the local file system and calculated the writing and reading speeds in megabytes per millisecond. The experiments are conducted on five structured data files of various sizes. The smallest data size was the same as the block size in HDFS, which is 64 MB. Two merged files are the minimum required. As a result, depending on the data size, we have selected two fragment sizes. We have chosen the same fragment size for data sizes 64 MB and 128 MB, which resulted in two and four merged files, respectively. However, we picked 64 MB as the fragment size for 256 MB, 512 MB, and 1024 MB.

2) THE PROPOSED FRAMEWORK COMPARISON WITH THE DEFAULT HDFS

In order to evaluate the overhead, we compared the proposed framework with the generic HDFS. The performance overhead of the writing experiments is represented by the speed differences of the generic HDFS and our implementations, as shown in Figure 8. The overhead imposed by the proposed fragmentation algorithm on a dataset is shown in Figure. 10. The overhead is shown in this Figure is the time taken by the proposed fragmentation algorithm on a dataset of various sizes ranging from (64MB to 1024 MB). There are five points in this Figure, and each point represents the results of data size. Also, it depicts the overhead imposed by the fragmentation algorithm applied on the low-sensitive data. Figure 11 shows the overhead imposed by the fragmentation & encryption algorithm used on high-sensitive datasets. Figures 10 and 11 show that the overhead for low-sensitive data fragmentation is lower than high-sensitive data fragmentation. Thus, applying encryption on only high-sensitive data will avoid a significant overhead imposed by encrypting the complete dataset. Similarly, the experimental results in the reading scenario are summarized in Figure 9, Figures 12 and 13. Figure 12 shows the overhead of the proposed defragmentation algorithm on different data sizes. It shows the overhead imposed on the low-sensitive data using a defragmentation algorithm. The overhead shown in Figure 13 is the time the proposed defragmentation algorithm takes on the same datasets used in previous scenarios. This Figure shows the overhead imposed on high-sensitive data using the defragmentation & decryption algorithm. Figures 12 and 13 illustrate that the overhead posed by defragmentation of the low-sensitive data is less than that of defragmentation & decryption of high-sensitive data. Tables 4 and 5 summarize the computation time of writing and reading several data sizes using our proposed techniques compared to HDFS. The comparison between Generic HDFS, fragmentation, fragmentation & encryption is shown in Table 4. The comparison between Generic HDFS, defragmentation, defragmentation & decryption is shown in Table 5.

TABLE 4. Data write experiment results.

Data Size	HDFS (ms)	Fragmentation (ms)	Fragmentation & Encryption (ms)
64MB	964	1023	1206
128MB	28930	29039	29230
256 MB	34672	34887	35312
512MB	87141	87657	87959
1024MB	21113	212069	212501

TABLE 5. Data read experiment results.

Data Size	HDFS (ms)	Defragmentation (ms)	Defragmentation & Decryption (ms)
64MB	4150	4185	4432
128MB	8558	8648	9038
256 MB	14380	14484	15170
512MB	26185	26591	27288
1024MB	52582	53785	54553

TABLE 6. Overhead ratio in writing experiments.

Data Size	Fragmentation& Encryption overhead ratio (%)	Fragmentation overhead ratio (%)
64MB	1.2510 (25.10)	1.0612 (6.12)
128MB	1.0104(1.04)	1.0038 (0.38)
256 MB	1.0185(1.85)	1.0062 (0.62)
512MB	1.0094(0.94)	1.0059 (0.59)
1024MB	1.0065(0.65)	1.0044 (0.44)

a: DISCUSSION ON THE WRITING EXPERIMENTS

The writing speeds in all experiments are summarized in Table 4. Consider the results of the generic HDFS as a baseline. The overhead of fragmentation has less overhead than fragmentation & encryption in all writing experiments. For analyzing the results, we computed the overhead ratio caused by the fragmentation and fragmentation &

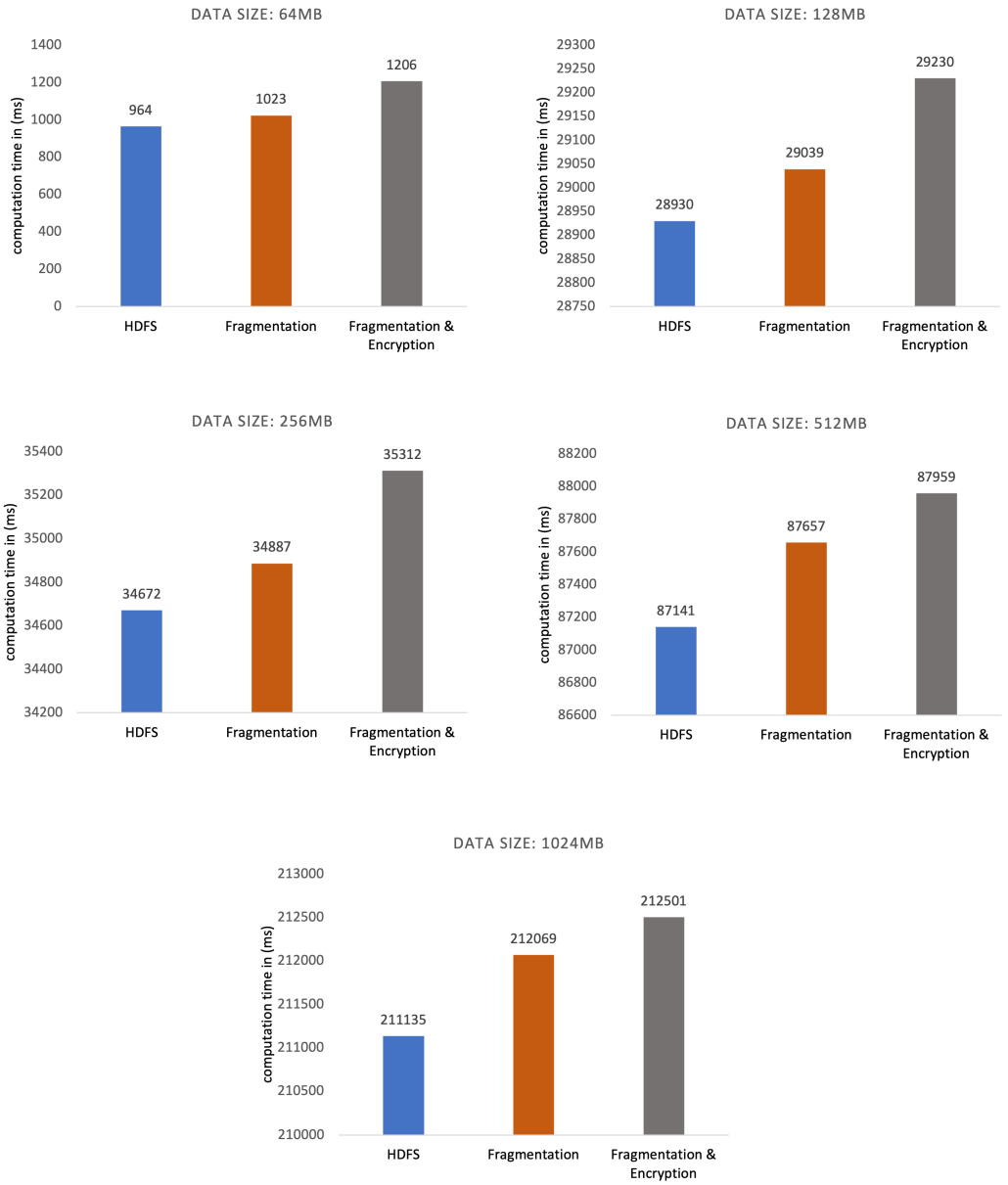

FIGURE 8. Performance comparison of proposed methods with HDFS for different data sizes.(a) 64MB, (b) 128MB,(c) 256MB,(d) 512MB, and (e) 1024MB.

encryption respective to the HDFS in the writing experiments. The overhead ratio is depicted in Figure 14. We found that for a 64MB data size, the overhead ratio was 1.0612 and 1.2510 for fragmentation, and fragmentation & encryption, respectively. When the file size is 1024MB, the overhead ratio decreases to 1.0044 and 1.0065 for fragmentation and fragmentation & encryption, respectively. Moreover, we observed that when we decrease the number of merged files, the fragmentation overhead decreases. The number of merged files depends on the fragment size. There is a trade-off in choosing the fragment size on performance as well as security. Smaller size fragments cause faster fragmentation but slower defragmentation. On the other hand, selecting a larger fragment makes it more vulnerable than a smaller one. We also noticed in the reading experiments that the larger fragments decreased the number of merged files, consequently reducing the overhead of the defragmentation and decryption processes. Our experiments and results found that the proposed techniques impose an additional 0.38%–1.85% overhead in writing experiments for data larger than 64MB in size. Table 6 illustrates the results of computing the overhead ratio when using different file sizes.

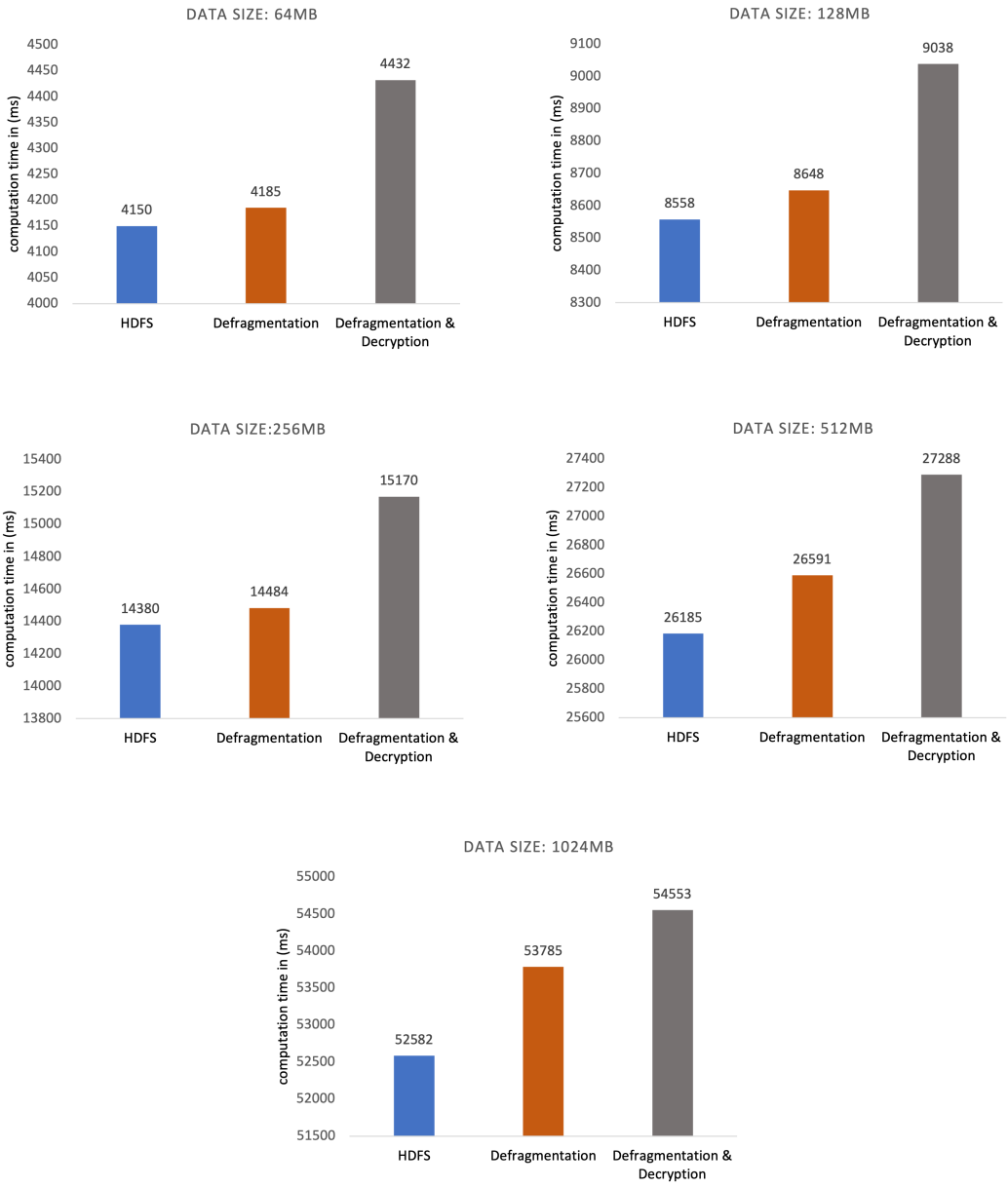

FIGURE 9. **Performance comparison of proposed methods with HDFS for different data sizes.(a) 64MB, (b) 128MB, (c) 256MB,(d) 512MB, and (e) 1024MB.**

b: DISCUSSION ON THE READING EXPERIMENTS

The reading speeds in all experiments are summarized in Table 5. We considered the results of the generic HDFS as a baseline. Defragmentation alone performs better than defragmentation combined with decryption in all reading experiments. As the tested data size increases in each experiment, so does the overhead. As a result, high overhead will be avoided upon user preferences for eliminating the encryption when not needed while maintaining a sufficient security mechanism for low-sensitive data. In the reading experiments, we calculated the overhead ratio caused by defragmentation and defragmentation & decryption techniques compared to the default HDFS. Figure 15 shows the overhead ratio. The overhead ratios for defragmentation and defragmentation & decryption, respectively, were 1.0084 and 1.0680 for a 64MB data size. Defragmentation and defragmentation & decryption's overhead ratios reached 1.0229 and 1.0375, respectively, in the case of 1024MB data size. The increased number of merged files in the 1024MB causes latency in the defragmentation process. On the other hand, the ratio in defragmentation & decryption technique has been reduced due to the use of parallel techniques that accelerate the decryption process. According to our findings and experiments, the proposed techniques add an additional 0.72%–5.6 % overhead when conducting reading experiments for datasets larger than

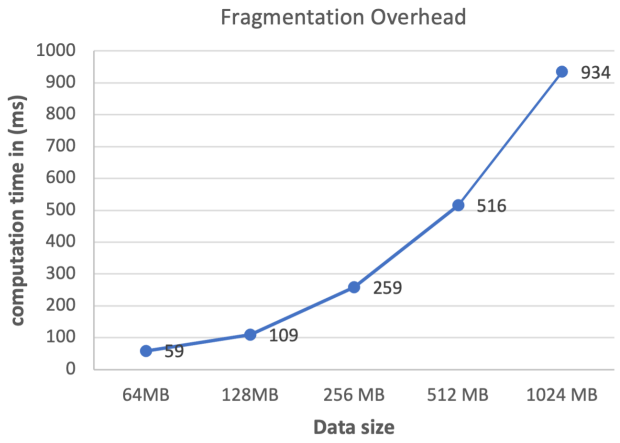

FIGURE 10. Fragmentation overhead.

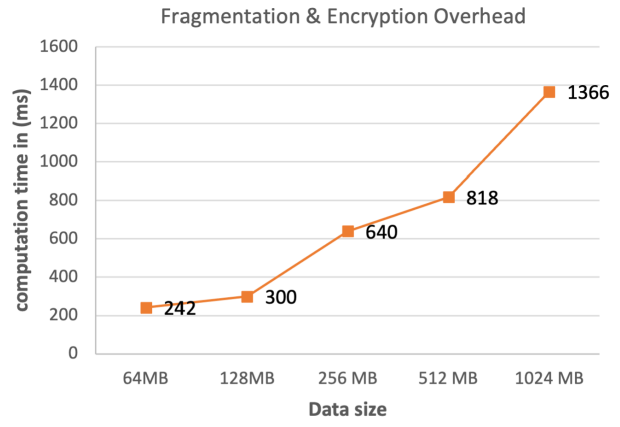

FIGURE 11. Fragmentation & encryption overhead.

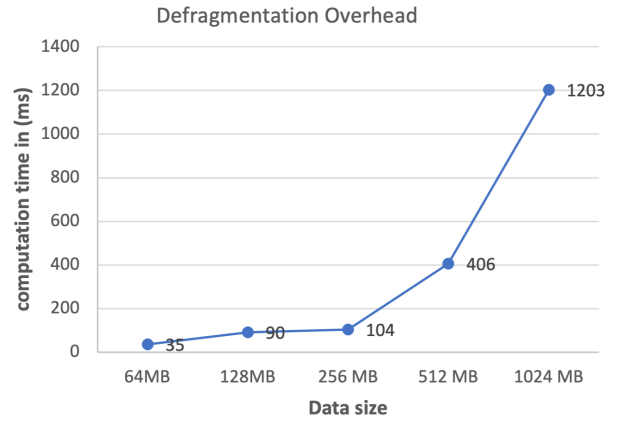

FIGURE 12. Defragmentation overhead.

64MB. A variety of data sizes was tested to determine the overhead ratio, the results of which are shown in Table 7.

TABLE 7. Overhead ratio in reading experiments.

Data Size	Defragmentation& Decryption overhead ratio (%)	Defragmentation overhead ratio (%)
64MB	1.0680 (6.80)	1.0084 (0.84)
128MB	1.0561(5.61)	1.0105 (1.05)
256 MB	1.0549(5.49)	1.0072 (0.72)
512MB	1.0421(4.21)	1.0155 (1.55)
1024MB	1.0375(3.75)	1.0229 (2.29)

TABLE 8. Security issues vs. our solutions.

Security Issues	Solutions
Authentication and authorization	(1) Access Control manages the process of authentication and authorization for data owner and user. (2) Multi-factor authentication. (3) Secure access permissions.
Communication	(1) All communications among involved entities are via an SSL/TLS secure channel.
Data Security	(1) Identify high/low sensitive and non-sensitive data. (2) Partitioning sensitive data over two phases. (3) Encrypt high sensitive data using a strong cipher. (4) Secure Metadata. (5) Hashing to check data integrity.
Auditing	(1) All access events will be logged. (2) The auditing logs will be secured.

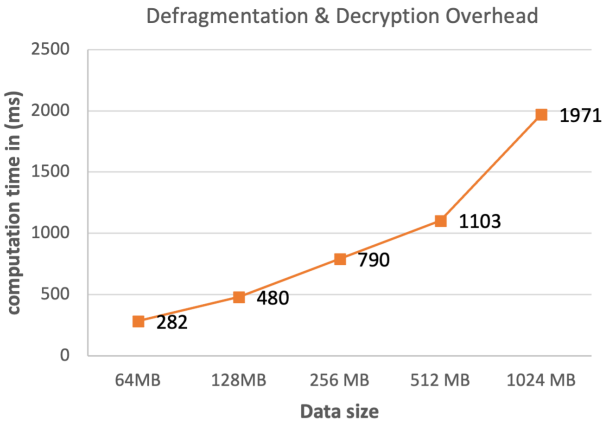

FIGURE 13. Defragmentation & decryption overhead.

VI. FRAMEWORK EVALUATION

Our proposed framework provides data security from end to end for general big data applications. Table 8 summarizes the security issues solved by our solutions. Furthermore, we have compared our proposed solutions with several

TABLE 9. Comparison of our framework with existing work.

Security Issues	Authentication & Authorization			Data Security					Communication	Auditing	
Solutions	(1)	(2)	(3)	(1)	(2)	(3)	(4)	(5)	(1)	(1)	(2)
Proposed framework	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[25]	✓	x	x	x	x	x	x	✓	x	✓	✓
[27]	✓	x	✓	x	x	x	x	x	x	x	✓
[30]	✓	x	✓	x	x	x	✓	x	x	✓	✓

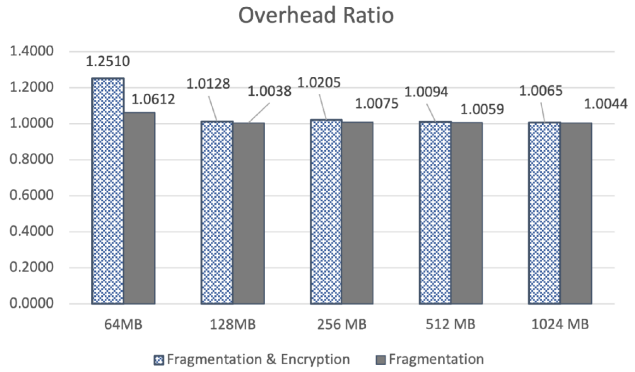

FIGURE 14. Overhead ratio in writing experiments.

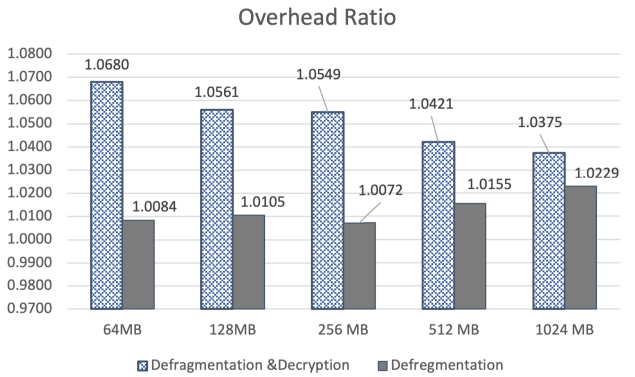

FIGURE 15. Overhead ratio in reading experiments.

blockchain-based schemes for big data in Table 9. The indices of the solutions in Table 8 are used as symbols referred to in Table 9.

VII. CONCLUSION AND FUTURE WORK

The security and privacy issues of big data systems are noteworthy and require demand attention. For instance, big data models such as Hadoop are built without any secure assumption. Furthermore, most of the existing tools rely on third parties, which imposes significant security issues. In this work, we proposed a big data security framework that provides data security using blockchain technology and fragmentation. The framework offers a secure environment for big data sharing, storage, and transmitting. Blockchain is responsible for providing the necessary security for big data storing and retrieving processes as well as access control and auditing mechanisms. Prior research has not adequately addressed big data security issues. For instance, previous studies have primarily focused on access control, data sharing, and auditing in specific big data domains such as smart homes and healthcare. However, our proposed framework is a generic solution that can be utilized in a wide range of big data domains. This study is still progressing, and the authors believe that additional details should be explored and published in a future paper. For future work, we are going to implement the complete scenario of our framework with blockchain technology. The Hyperledger fabric [40] platform will be used to implement the solution, which is a permissioned blockchain with higher transaction throughput and security than other blockchain platforms [41], [42].

REFERENCES

[1] M. Zwolenski and L. Weatherill, "The digital universe: Rich data and the increasing value of the Internet of Things," *J. Telecommun. Digit. Economy*, vol. 2, no. 3, pp. 1–47, 2014.
[2] X. Jin, B. W. Wah, X. Cheng, and Y. Wang, "Significance and challenges of big data research," *Big Data Res.*, vol. 2, no. 2, pp. 59–64, Jun. 2015.
[3] D. Lv, S. Zhu, H. Xu, and R. Liu, "A review of big data security and privacy protection technology," in *Proc. IEEE 18th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2018, pp. 1082–1091.
[4] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. 44th Symp. Theory Comput.*, 2012, pp. 1219–1234.
[5] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.
[6] R. Norvill, M. Steichen, W. M. Shbair, and R. State, "Demo: Blockchain for the simplification and automation of KYC result sharing," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 9–10.
[7] M. Kassen, "Blockchain and E-government innovation: Automation of public information processes," *Inf. Syst.*, vol. 103, Jan. 2022, Art. no. 101862.
[8] R. von Solms and J. van Niekerk, "From information security to cyber security," *Comput. Secur.*, vol. 38, pp. 97–102, Oct. 2013.
[9] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services," in *Proc. CIDR*, 2005, pp. 1–5.
[10] N. Santos and G. L. Masala, "Big data security on cloud servers using data fragmentation technique and NoSQL database," in *Proc. Int. Conf. Intell. Interact. Multimedia Syst. Services*. Cham, Switzerland: Springer, 2018, pp. 5–13.
[11] G. Memmi, K. Kapusta, and H. Qiu, "Data protection: Combining fragmentation, encryption, and dispersion," in *Proc. Int. Conf. Cyber Secur. Smart Cities, Ind. Control Syst. Commun. (SSIC)*, Aug. 2015, pp. 1–9.
[12] H. Heni, M. B. Abdallah, and F. Gargouri, "Combining fragmentation and encryption to ensure big data at rest security," in *Proc. Int. Conf. Hybrid Intell. Syst.* Cham, Switzerland: Springer, 2017, pp. 177–185.
[13] K. Kapusta and G. Memmi, "A fast fragmentation algorithm for data protection in a multi-cloud environment," 2018, *arXiv:1804.01886*.
[14] D. E. Bakken, R. Rarameswaran, D. M. Blough, A. A. Franz, and T. J. Palmer, "Data obfuscation: Anonymity and desensitization of usable data sets," *IEEE Security Privacy*, vol. 2, no. 6, pp. 34–41, Nov. 2004.

[15] C. Tankard, "Encryption as the cornerstone of big data security," *Netw. Secur.*, vol. 2017, no. 3, pp. 5–7, Mar. 2017.
[16] D. Lopez and B. Farooq, "A blockchain framework for smart mobility," in *Proc. IEEE Int. Smart Cities Conf. (ISC2)*, Sep. 2018, pp. 1–7.
[17] N. Deepa, Q. Pham, D. C. Nguyen, S. Bhattacharya, P. B. T. R. Gadekallu, P. K. R. Maddikunta, F. Fang, and P. N. Pathirana, "A survey on blockchain for big data: Approaches, opportunities, and future directions," *CoRR*, vol. abs/2009.00858, pp. 1–4, Oct. 2020.
[18] J. Anuradha, "A brief introduction on big data 5VS characteristics and Hadoop technology," *Proc. Comput. Sci.*, vol. 48, pp. 319–324, Jan. 2015.
[19] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016.
[20] A. Chakravorty and C. Rong, "Ushare: User controlled social media based on blockchain," in *Proc. 11th Int. Conf. Ubiquitous Inf. Manage. Commun.*, Jan. 2017, pp. 1–6.
[21] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–5.
[22] M. Castro and B. Liskov, "A correctness proof for a practical byzantine-fault-tolerant replication algorithm," MIT Lab. Comput. Sci., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TM-590, 1999.
[23] P.-L. Aublin, S. B. Mokhtar, and V. Quema, "RBFT: Redundant byzantine fault tolerance," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst.*, Jul. 2013, pp. 297–306.
[24] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.
[25] N. Gupta, A. Jha, and P. Roy, "Adopting blockchain technology for electronic health record interoperability," Cognizant Technol. Solutions, Teaneck, NJ, USA, White Paper, 2016. [Online]. Available: https://dokumen.tips/documents/adopting-blockchain-technology-for-electronic-health-record-.html
[26] C. Xu, K. Wang, G. Xu, P. Li, S. Guo, and J. Luo, "Making big data open in collaborative edges: A blockchain-based framework with reduced resource requirements," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[27] H. Es-Samaali, A. Outchakoucht, and J. P. Leroy, "A blockchain-based access control for big data," *Int. J. Comput. Netw. Commun. Secur.*, vol. 5, no. 7, p. 137, 2017.
[28] H. Yu, Z. Yang, and R. O. Sinnott, "Decentralized big data auditing for smart city environments leveraging blockchain technology," *IEEE Access*, vol. 7, pp. 6288–6296, 2019.
[29] J. Li, J. Wu, G. Jiang, and T. Srikanthan, "Blockchain-based public auditing for big data in cloud storage," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102382.
[30] C. Zhang, Y. Li, W. Sun, and S. Guan, "Blockchain based big data security protection scheme," in *Proc. IEEE 5th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Jun. 2020, pp. 574–578.
[31] Z. Guan, Y. Zhao, D. Li, and J. Liu, "TBDCT: A framework of trusted big data collection and trade system based on blockchain and TSM," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Sep. 2018, pp. 77–83.
[32] S. Li, "Application of blockchain technology in smart city infrastructure," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2018, p. 276.
[33] L. Yue, H. Junqin, Q. Shengzhi, and W. Ruijin, "Big data model of security sharing based on blockchain," in *Proc. 3rd Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Aug. 2017, pp. 117–121.
[34] B. K. Adhikari, W. Zuo, R. Maharjan, X. Han, and S. Liang, "Detection of sensitive data to counter global terrorism," *Appl. Sci.*, vol. 10, no. 1, p. 182, Dec. 2019.
[35] G. Xu, C. Qi, H. Yu, S. Xu, C. Zhao, and J. Yuan, "Detecting sensitive information of unstructured text using convolutional neural network," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2019, pp. 474–479.
[36] Z. Yang and Z. Liang, "Automated identification of sensitive data from implicit user specification," *Cybersecurity*, vol. 1, no. 1, pp. 1–15, Dec. 2018.
[37] *CIS Ubuntu Linux 18.04 BenchmarksT Resources*. Accessed: Oct. 30, 2021. [Online]. Available: https://www.cisecurity.org/cis-benchmarks
[38] G. Singh and S. Supriya, "A study of encryption algorithms (RSA, DES, 3DES and AES) for information security," *Int. J. Comput. Appl.*, vol. 67, no. 19, pp. 33–38, Apr. 2013.
[39] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, and S. McLachlan, "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record," *J. Amer. Med. Inform. Assoc.*, vol. 25, no. 3, pp. 230–238, Mar. 2018.
[40] E. Androulaki, A. Barger, V. Bortnikov, and C. Cachin, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15.
[41] A. Manzoor, A. Braeken, S. S. Kanhere, M. Ylianttila, and M. Liyanage, "Proxy re-encryption enabled secure and anonymous IoT data sharing platform based on blockchain," *J. Netw. Comput. Appl.*, vol. 176, Feb. 2021, Art. no. 102917.
[42] I. Makhdoom, M. Abolhasan, and W. Ni, "Blockchain for IoT: The challenges and a way forward," in *Proc. 15th Int. Joint Conf. E-Bus. Telecommun.*, 2018, pp. 1–12.

HANAN E. ALHAZMI received the B.S. degree in computer science from Umm Al-Qura University, Saudi Arabia, in 2008, and the M.S. degree in computer sciences from King Abdulaziz University, Saudi Arabia, in 2013, where she is currently pursuing the Ph.D. degree with the Department of Computer Science. She is also working as a Lecturer at Umm Al-Qura University. Her current research interests include security engineering, blockchain, big data, the IoT, and computer networking.

FATHY E. EASSA received the B.Sc. degree in electronics and electrical communication engineering from Cairo University, Egypt, in 1978, the M.Sc. degree in computers and Systems engineering from Al-Azhar University, Cairo, Egypt, in 1984, and the Ph.D. degree in computers and systems engineering from Al-Azhar University, joint supervision with the University of Colorado, USA, in 1989. He is currently a Full Professor with the Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia. His research interests include agent-based software engineering, the IoT security, software engineering, big data management and security, distributed systems security, and exascale systems testing.

SUHELAH M. SANDOKJI received the B.Sc., M.Sc., and Ph.D. degrees in computer science from King Abdulaziz University, Jeddah, Saudi Arabia, in 2006, 2012, and 2019, respectively. She currently holds a teaching position at the Information Technology Department, Faculty of Computing and Information technology, King Abdulaziz University. Her research interests include software engineering, specifically testing software agent, HPC computing and data parallelism, task scheduling, GPGPU as an accelerator, heterogeneous computing, big data, using blockchain for security in big data, and the IoT.