INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE

ISSN: 2057-5688

An Innovative Framework for Safe Data Sharing and Authentication in a Cloud-Based Big Data Setting

BARUPATI VINOD KUMAR
Research Scholar
Department of Computer Science
J.S UNIVERSITY, Shikohabad, UP

Dr. VIKRAM SINGH RATHORE
Professor
Department of Computer Science
J.S. UNIVERSITY, Shikohabad, UP

Dr. VITTAPU MANI SHARMA
Professor
Department of Computer Science
J.S. UNIVERSITY, Shikohabad, UP

ABSTRACT

Cloud big data security is becoming more complex as the number of data sources continues to grow at a fast pace. Concerns around data management, integrity, privacy, and infrastructure security are all part of Big Data security. Cloud computing has made big data analytics, storage, and processing possible, however the cryptographic methods used to secure large data on the cloud are inadequate. In this paper, we outlined the main issues with cloud-based big data security and offered a general framework for addressing them. Our proposed novel system architecture is dubbed SADS-Cloud, which stands for Secure Authentication and Data Sharing in Cloud and is ideal for use in a Big Data Environment. I outsourcing large data, (ii) sharing big data, and (iii) managing big data are the three procedures covered in this paper. When it comes to large data outsourcing, the SHA-3 Hashing Algorithm is employed for data owner registration to the Trusted Center. The MapReduce technique is used to split the input file into chunks of a predetermined size. The SALSA20 algorithm is used to encrypt each block. Participants in data exchange engage in secure file retrieval. User credentials (i.e., ID, password, secure ID, current timestamp, and email ID) are hashed and compared to the database in order to achieve this. In order to organize massive data in the following ways, big data management employs three essential procedures: A number of algorithms, including Fractal Index Tree, Density-based Clustering of Applications with Noise (DBSCAN), and Lemperl Ziv Markow Algorithm (LZMA), are used to compress and index the files stored in the cloud database. This paves the way for personalized searches, as well as additions and removals. The proposed system was tested using the following metrics: throughput, compression ratio, information loss, encryption time, and decryption time; Java programming was used for its implementation.

Keywords:
Big Data Outsourcing, Big Data Sharing, Big Data Management, SALSA Encryption with MapReduce, Fractal Index Tree, and SHA-3

1. INTRODUCTION

Distributed storage has made on-demand data sharing feasible by improving digital data preservation, retrieval, and distribution. Information security is a major concern that limits many cloud applications. However, simple and speedy access might pose a security risk when exchanging sensitive data [(Huang et al. 2015)]. Cloud admins' easy access to sensitive data raises security concerns. Consumer anxiety and disapproval of distributed computing in

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

banking and government agencies increase due to this concern. The conventional Big Data strategy does not highlight organized and unstructured data sensitivity. Insurance and security should be included to reduce the risk of personal information disclosure. Due to the large amount of data and the mix of structured and unstructured information, new Big Data models are needed to improve security. Academics are increasingly interested in Big Data security because hackers target Big Data storage systems [(Santos and Masala 2019)], [(Jeong and Shin 2016)]. Volume, velocity, and diversity are the three pillars of "Big Data". Big Data's practical applications are shown in [(Bronson, K., & Knezevic 2016)], [(Bholat 2015)], and [(Bearman 2015)]. Big Data storage systems' complexity, security breaches, and large data volumes make it difficult to address these issues in the real world [(Meshram et al. 2019)]. It's hard to trust cloud or huge data servers. Tensor rules are used to compute blend arithmetic operations over real numbers in the Fully Homomorphic Encryption for Blend Operations (FHE-BO) model [Gai and Qiu 2018]. CEA (Cyber-Enabled Applications) generates optimal task assignment plans for energy-aware green computing. Gai et al. (2016) addresses energy waste in dynamic networking, and [(Gai, Qiu, and Zhao 2018)] reduces energy usage. Healthcare will be our focus. Healthcare applications need a large, secure server to store 140 GB of genome data [(Stergiou and Psannis 2017)]. Security breaches on cloud and big data servers expose data owners' sensitive information [(Wei et al. 2019)]. Brute-force, stolen verifier, password guessing, and other security breaches may harm Big Data storage systems. Encrypting data transfers using ciphertext has failed to preserve users' and owners' privacy and confidentiality [(Li et al. 2017)]. Big Data worries about data security. Considering the importance of security, it is necessary because:

- Access policy is not designated when ciphertext is updated and here user legitimacy is failed that means who intends to access the data are still a great concern in Big data
- There is no authorized entity to monitor the data sharing and outsourcing to storage systems.

Big data storage system authentication, confidentiality, and integrity must be monitored constantly. Real-time usage include smart grid, transaction, and e-healthcare [(Jain, Gyanchandani, and Khare 2019)]. Encryption is the key to permission and data security [(Ma 2018)]. When much data is aggregated, less storage space is required. Big data security should be considered while grouping data types. Big Data clustering poses dangers, including data loss [(Lighari and Hussain 2018)]. The encryption procedure after data compression is the main reason [(Castiglione et al. 2015)]. Compression before encryption can fix such errors. Many safe clustering approaches have been proposed. Wang and Lin (2016) recommend Locality Sensitive Hashing (LSH) for similarity clustering. It

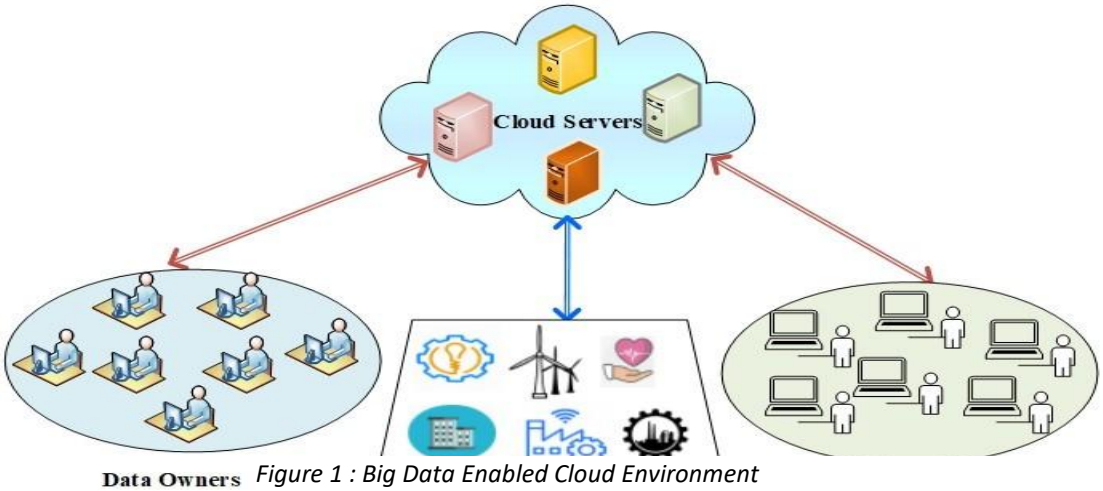

Figure 1 : Big Data Enabled Cloud Environment

ISSN: 2057-5688

calculates dataset similarity. LSH, which works well with smaller datasets, cannot find similarities in Big Data's terabytes, gigabytes, and petabytes. Due to data volume, clustering efficiency is crucial [(El Hadj et al. 2019)]. Big data enabled cloud environments solve the challenge of inexpensively storing enormous volumes of data. Fig. 1 shows the Big Data-enabled Cloud for data owners and users. Uses include apps and services. When addressing Data

What we found makes it sad that we have to point out that none of the present encryption methods offer enough security, and all of them have the same problem: they take a long time to compute. Our main goal is to make an encryption system that is easy to use and can be proven to be safe. The main idea of the paper is to suggest a new way to make big data settings safe for authentication and sharing data in the cloud. The name for this method is SADS-Cloud. The rest of the document's parts are grouped in this way: In Section 2, the problems that this field of study is facing are thoroughly explained. In Section 3, these problems are brought to light. The suggested way to deal with the problems listed above was talked about in SECTION 4. Section 5 goes into detail about the benefits of the suggested system when it is put into action. It also shows the results of the tests and compares them to current methods. This part also helps you see where earlier attempts fell short. In Section 6, the paper's conclusion and suggestions for how it could be improved in the future are summed up. This section also suggests ways that the paper could be improved.

2. Related Work

In this section, we cover the state-of-the-art of Big data security over Cloud environment. We categorize this section into two classes i.e. User Privacy (Authentication) in Big data Cloud and data security and retrieval in the Big data Cloud.

Owners, data quality, security, and responsibility and accountability are typical words. Data owners store their huge data on the cloud, which is straightforward and effective. Data consumers are those whose occupations demand data consumption. Services and applications can be from healthcare, manufacturing, renewable energy, building management, energy management, and more.

In a hierarchical attribute authorization structure, [Shen et al., 2017] proposed a secure authentication method that makes use of a tree-based signature. It is used in a multi-tiered authentication system. Protecting against replay and forgery attacks, it ensures privacy preservation. An authorized hierarchical structure of attributes increases the time complexity and storage difficulties. A secure communication protocol developed to monitor the system's functioning, [(Aditham and Ranganathan 2018)] established the two-step attack detection approach. Every procedure begins with the building management of the system. In the second step, replica nodes are matched with instructions. In safe data transfer, user and data privacy are jeopardized by randomly generated keys by data nodes. We have already covered cloud access management and massive data processing in [(Reddy 2018)]. This work made use of access control for data access. One alternative to using symmetric keys for authentication is Kerberos, which relies on a third party. It opens up info that isn't secure. This research integrates anomaly and access control to safeguard data from malicious users. Data monitoring and control are two of Spike's primary functions. This work does not support a variety of data sources, such as text, images, audio, or video. [(Vorugunti 2017)] has developed a new framework called PPMUS, which stands for privacy-preserving mobile user authentication system. Its used Big Data qualities include storage capacity, user-friendliness, and strong Big Data management. Fuzzy Hashing and

ISSN: 2057-5688

Fully Homomorphic Encryption (FHE) are the two methods that are used to safeguard user privacy. In order to construct the user profile, fuzzy hashing and FHE are used. Its creation takes place in the cloud, where servers and users communicate online. One major issue with this architecture is that it attempts to determine the user's authenticity or fraudsteriness based on how they type their password. On the other hand, this method does not protect user privacy since dishonest people may use the same ideas to steal information. [(Zhao, Li, and Jiang 2018)] laid forth a secure authentication method for users that relies on passwords in scenarios when there are several servers. For user authentication, we choose elliptic curve cryptography (ECC), which prevents impersonation and offline password guessing. Efficiency and security were the metrics used to evaluate the results of the experiments. In addition, the proposed strategy is effective for use in real-time scenarios. The ECC asymmetric cryptography method generates smaller key sets for both public and private use. The computation and transmission cost is high, which is a major drawback when compared to hashing methods.

3. Problem Statement

Traditional approaches relied on three main tenets: data security, user privacy, and retrieval. The proposed strategy incorporates all three classes. The policy of action in [(Hu et al., 2018)] was created by data owners who securely updated their credentials and original data. To update a new access policy in a secure and verifiable manner, an upgraded NTRU cryptosystem was proposed. Wrap Failure and Grap Failure, two problems with older NTRU systems, are addressed. Furthermore, secret sharing was shown using (b,n) thresholding. Cloud access control systems are vulnerable to attacks. When a data owner is compromised, the hacker gains full access to private data. So, a reliable third party is required for data owners' governance and regulation. In order to address both the clustering problem and the need for data security, [(Nafis and Biswas 2019)] used a multidimensional clustering approach in conjunction with the simple data encryption standard (SDES). In order to reduce data size and sidestep heavy overheads, the Huffman compression (HC) method was created. But, the best practice is to compress the source file before encrypting it. This article failed to find a way to decrease computing time by compressing and encrypting data. It performs well with relatively small datasets. As a very simplistic encryption scheme, SDES is severely lacking in security. (Ramya Devi and Vijaya Chamundeeswari, 2018) proposed using triple DES in healthcare settings. Anonymization, data encryption (using triple DES), and authentication (using SHA-256) are the three operations that comprise this research. The main problems with this study are as follows: (1). The encryption and decryption durations for Triple DES are substantial, even for small blocks (64 bits) (2). The security of user data exchanges might be jeopardized if insufficient attributes are considered during authentication. Replay, stolen verifier, denial-of-service, chosen plaintext, server size compromise, man-in-the-middle attack, and other security threats were used in [(Chattaraj et al. 2018)] HEAP. The security measures used in this research are AES and ECC. Both the encryption and decryption times are increased for the massive amounts of data when the two methods are combined (AE S and ECC).

4. Proposed Work

In this work, we created a new system architecture dubbed SADS-Cloud—which stands for Secure Authentication and Data Sharing in Cloud-enabled Big Data Environment—to fix the problems with the existing techniques. The major objective of this study is to provide a method for securely storing and retrieving large datasets in a cloud-based

ISSN: 2057-5688

environment.

4.1 Big Data Outsourcing from Data Owners

For data outsourcing to the CS, data owners must register their identities to the TC. There are three steps are involving in this stage:

(1). *Registration:* In this step, the data owner registers to TC by following identities: Mail ID, $User_{ID}$, Password, Current Timestamp and Secure ID. User ID and Password are hashed and then given to TC for registration. After registration, TC generates a hash value for DO given information using SHA3-384. The details of the registration process are shown in Fig.3.

(2). *Login:* When DO enters for login to CS. They must be providing valid information such as $User_{ID}$, Password, Current Timestamp and Secure ID. Then wait for successful authentication response CS.

(3). *Authentication:* For login purposes, all identities are hashed and stored in the database. Again it is hashed to compare the database with the given DO information. The step-wise description of user authentication is shown in Fig.4.

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

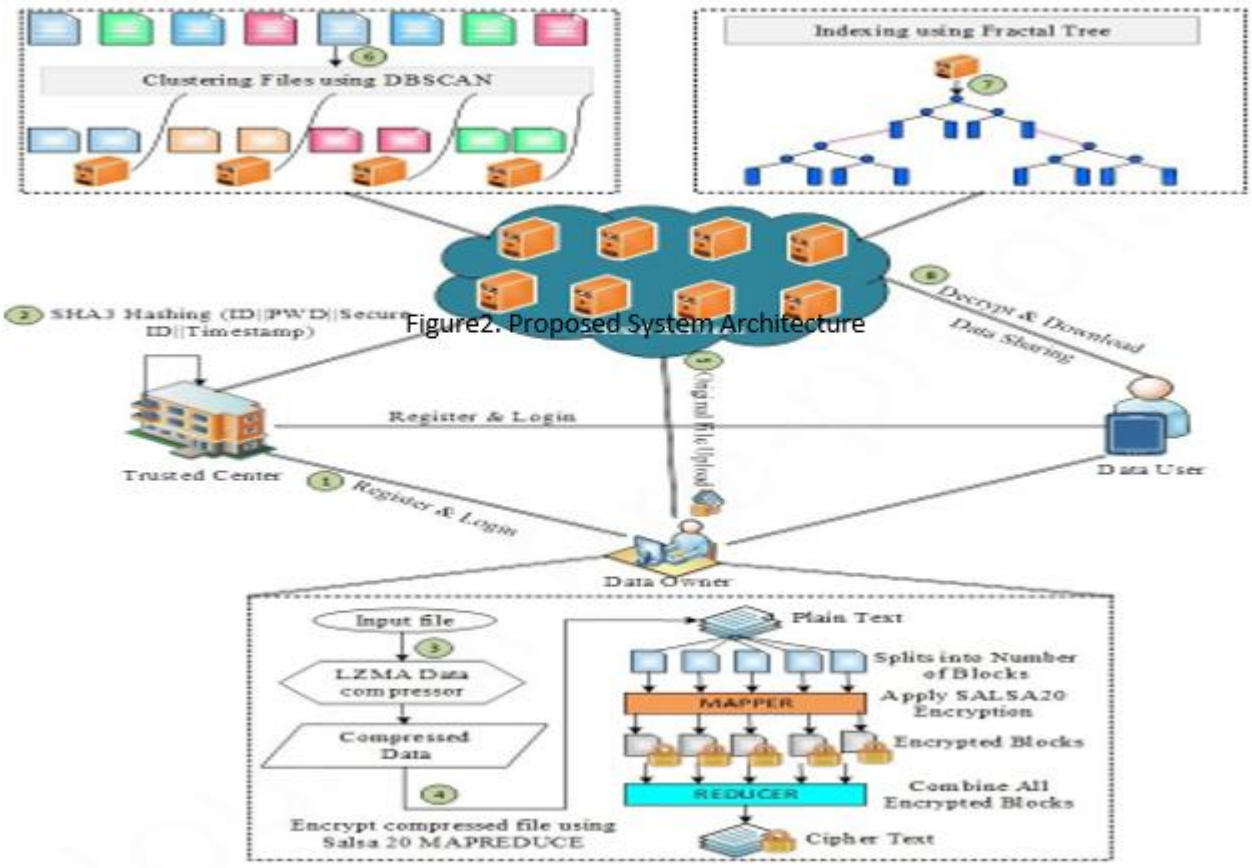

Figure2. Proposed System Architecture

Figure 2. Proposed System Architecture

Upon TC authentication success, DO will ask for the private key to encrypt data. The DO specifies the desired degree of security, and TC creates a private key accordingly. For DO, there are three distinct sensitivity levels: (i). The second one is sensitive. Most Sensitive and (ii). Without emotion. On the other hand, sensitive data needs access controls. Therefore, in order to prevent security breaches, the CS monitors the quantity of data accesses. TC uses the SALSA 20 Encryption technique to produce keys. Data classified as sensitive has a key size of 20–128 bits, whereas data classified as very sensitive uses a key size of 256 bits. Outsourcing the encryption of huge data sets to the cloud is a time-consuming procedure since both the encryption and decryption phases require a long time. The SALSA 20 Encryption-MapReduce system is being studied as a solution to these problems.

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

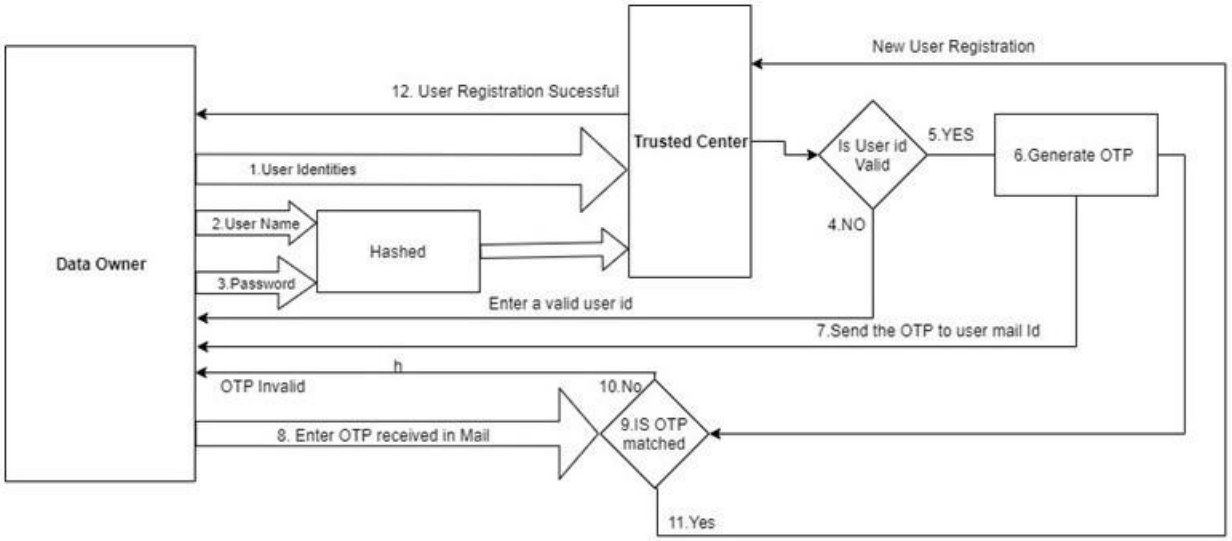

Figure 3. Data User Registration with Trusted Center

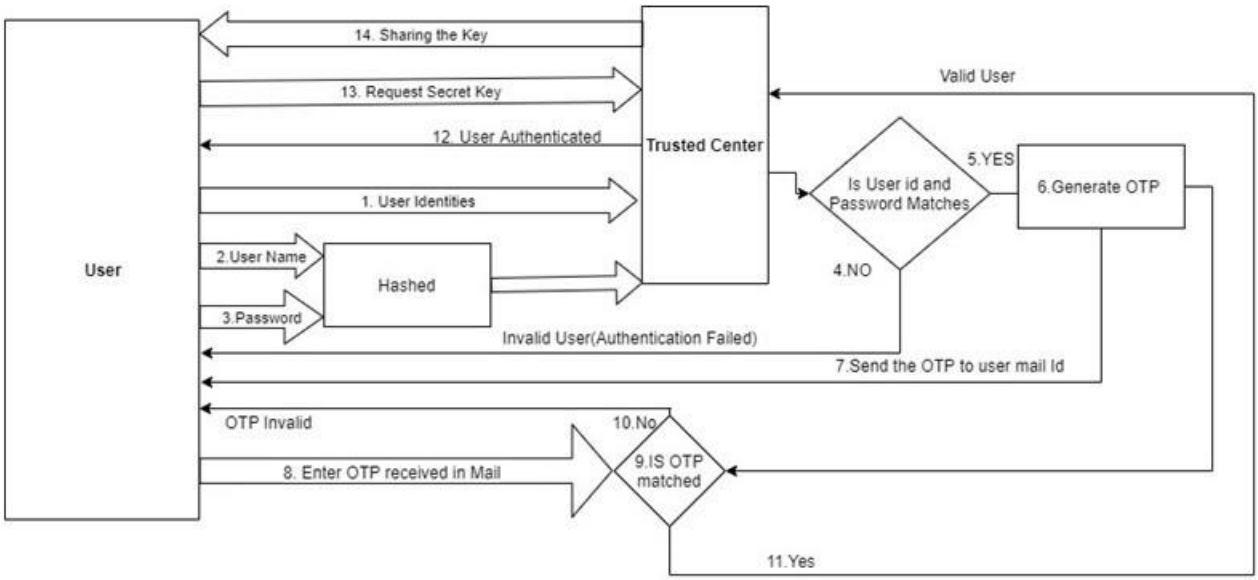

Figure 4. User Authentication with Trusted Center

4.1.1 *SHA3 Hashing Algorithm*: It is a secure hashing algorithm, which composed of four hashing functions include SHA3- 224, SHA3-256, SHA3-384, and SHA3-512. It also consists of two Extendable Output Functions (XOFs) include SHAKE-128, SHAKE256. The SHA-3 hashing algorithm is proposed for message authentication in registration. It is based on the Keccak algorithm with Sponge Construction. The procedure for SHA-3 hashing algorithm is as follows:

PSEUDOCODE FOR SHA-3 HASHING ALGORITHM

Step 1) Begin

Step 2) **SHA3:**= **PROC**(M::STRING, MT::Name:=TEXT)

ISSN: 2057-5688

Step 3) **Local** , , ;

Step 4) **If** Type (PROCname,"INDEXED")

Step 5**)** **Then** N:= OP('PROCname')

Step 6) **Else**

Step 7) **Error** 'Output length is not specified'

Step 8) **End If;**

Step 9) **If Not** in Output (224, 256, 384, 512) Then

Step 10) **Error** 'Not a Valid Output Length', N

Step 11) **End If;**

Step 12) M:= MessagetoBytes (M, MT);

Step 13) := Keccak (M, 1600, 1600-2.N, N, Hash);

Step 14) ( );

Step 15) EndPRO

In hashing, the input is padding functions which are the messages provided in list of integers or bytes range from 0 to 255, Domain and Bit Rate. In domain, it considers the Hash, XOF and KEC, which require various paddings and run Domain Separation by differentiating input and corresponding to a hash function. Finally, an output for this padding function is to the array, which contains padding message blocks and each and every block consists of list of integers. The above procedure is implemented for hashing for secure user authentication at both TC and CS. PROC indicates the procedure, N is the output length bits (224, 256, 384, and 512). M is the message and MT is the message type. The whole procedure is executed for hashing DO and DU information

4.2.2. LZMA for Data Compression: Firstly we compress the data before data encryption. To mitigate the issues of Huffman compression, in this work we proposed LZMA (Lempel Ziv Markow Algorithm) for compressing data. After that, we perform data encryption based on the aforementioned procedure. In LZMA, delta encoder and sliding dictionary encoder with LZ77 dynamic dictionary encoding are proposed and the output of LZMA encoder is presented in terms of tuples include Offset, Length and New Symbol. The functioning of delta encoder and decoder is the following:

- Delta Encoder: It forms the input data for compression using the Sliding Window. It stores and transmits data in sequential forms.
- Delta Decoder**:** The outcome of this entity is to keep the first data stream and subsequent bytes are stored based on performing addition operation between current data byte and preceding data byte.

The LZMA are source coding calculations which differ from those that we have recently examined (for example Huffman Codes and Shannon Codes) in the following ways: (i) They utilize variable-to-variable-length codes, in which both the quantity of source symbol encoded and the quantity of encoded bits per codeword are variable. Also, the code is time-varying. (ii) They don't require earlier information on the source insights, yet after some time they adjust with the goal that the normal codeword length L per source letter is limited. Such an algorithms called universal. (iii) They have been broadly utilized in practically speaking; in spite of the fact that more up to date plots enhance them, they give a straightforward way to deal with understanding all-inclusive information about universal data compression algorithms. (iv)Lossless pressure is commonly utilized for applications that can't endure any difference between the first and reproduced information. (v) It uses a dictionary compression scheme and features a high

ISSN: 2057-5688

compression ratio and a variable compression-dictionary size. (vi)Small memory requirements for decompressing (depends on dictionary size). (vii) Supporting multi-threading.

4.2.3. SALSA20 with MapReduce: We encrypt compressed data for preserving the security of data in CS. For that, we proposed the SALSA20 algorithm with the MapReduce paradigm. It is an encryption algorithm provides several

benefits to other symmetric algorithms. We firstly describe the MapReduce model and the SALSA20 encryption algorithm.

STEPS INVOLVING IN THE CONFIGURATION FOR EXPERIMENTS ARE THE FOLLOWING

Step 1)	Ubuntu 14.04LTS installed every node
Step 2)	JDK 1.8.0 installed in every node
Step 3)	Hadoop 2.7.2 installed in all nodes
Step 4)	Nodes in Hadoop, categorized into Master and Slave nodes
Step 5)	Create configuration and make connectivity for communication

Table 1. System Configuration

Hardware (Intel i5 Core Components)	Processor	3.30GHz
	CPU	4Cores
	Memory	8GB
	Hard Disk	500GB
	Connectivity	Gigabit Ethernet LAN
Software (Single Node Cluster)	Operating System	Ubuntu 14.04LTS
	JDK	1.8.0
	Hadoop	2.7.2
	Netbeans	8.2

1.1 Comparison Study

In this section, a comparative study of the proposed scheme and previous similar approaches is presented. Information loss, compression ratio, throughput, encryption time, decryption time, and efficiency are considered as the evaluation metrics for comparison between the proposed and previous approaches include SDES with HC and Triple DES.

5. EXPERIMENTAL RESULT

In this section, we cover the experimental results with three sub-sections include experimental setup, comparison study, and results and summary for the proposed SADS-Cloud and previous schemes.

5.1 Experimental Setup

Table 3 shows the drawbacks of existing methods.

ISSN: 2057-5688

Table.3. Drawbacks of Existing Approaches

State-of-the- art	Contributions	Drawback
SDES with HC	(1). Implement SDES for file encryption (2). Apply HC for encrypted data compression (3). Error control on compressed file (4). Use clustering on error controlled data	• Low-level security • High computation time • Suitable for small-sized data
Triple DES	(1). User Authentication using SHA-256 (2). Triple DES for encryption (3). Anonymization	• Lack user-privacy • Poor Scalability • Time-consuming

5.2.1. Information Loss: It is also known as "Data Quality". Information loss is a significant metric for data retrieval in Cloud. During encryption/compression, we must use some error conditions to recover the data from failures. Otherwise, information is lost at any stage. In addition, information is due to any malicious actions over a Cloud database. Information loss for the proposed and previous approaches can be seen in Fig.7 the performance of the proposed SADS-Cloud is compared with SDES with HC and Triple DES. When compared to SDEA with HC and Triple DES, our proposed scheme shows better results i.e. information loss of our proposed scheme is very small. Despite the LZMA compression technique, we obtained very small information loss. In SDES with HC, compression is invoked after encrypting data and then error conditions are verified and applied over encrypted data. In triple DES, information loss is more since there is no compression scheme is invoked. The average information loss for the proposed SADS Cloud is 0.023% whereas SDES with HC and Triple-DES is 0.07%, and 0.175%, respectively.

ISSN: 2057-5688

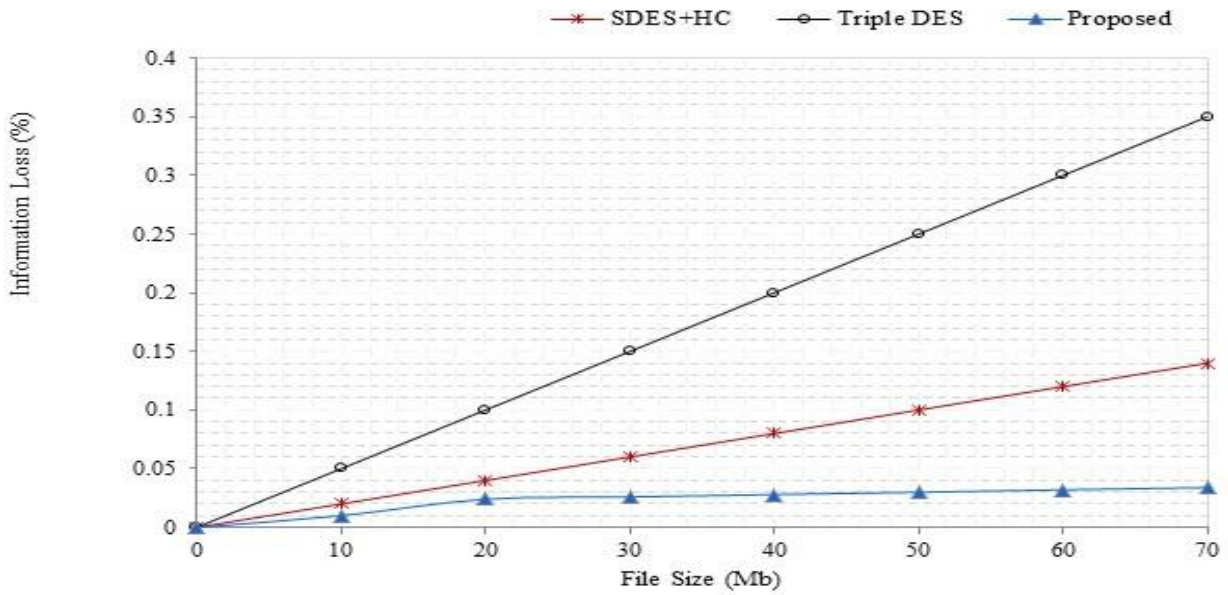

Figure 5. Comparison of Information Loss vs. File Size

5.2.2. Compression Ratio: The compression ratio is a metric that plays a significant role in the Big data-enabled Cloud. It is defined as the rate of data after performing the compression. However, data compression techniques can be classified into two categories such as lossy compression and lossless compression. Lossless compression techniques are beneficial than lossy compression since it does not reduce original contents. It is calculated as follows:

$$Compression\ Ratio = \frac{CFS}{OFS\ before\ Compression} \quad (1)$$

Where is the compressed file size, and is the original file. Fig 8 indicates the compression ratio performance of the proposed SADS-Cloud and SDES with HC.

If any of the compression techniques show high CR, then it means that the technique is poor to minimize the original data. When compared to SDES with HC, our proposed SADS-Cloud scheme produces very less compression ratio. With the use of the LZMA compression technique, our proposed SADS-Cloud has provided a high compression ratio. In SDES with HC, the Huffman compression technique is proposed which ensemble changes according to frequencies and probabilities, but it does not consider Block of Symbols and it cannot be practical for large

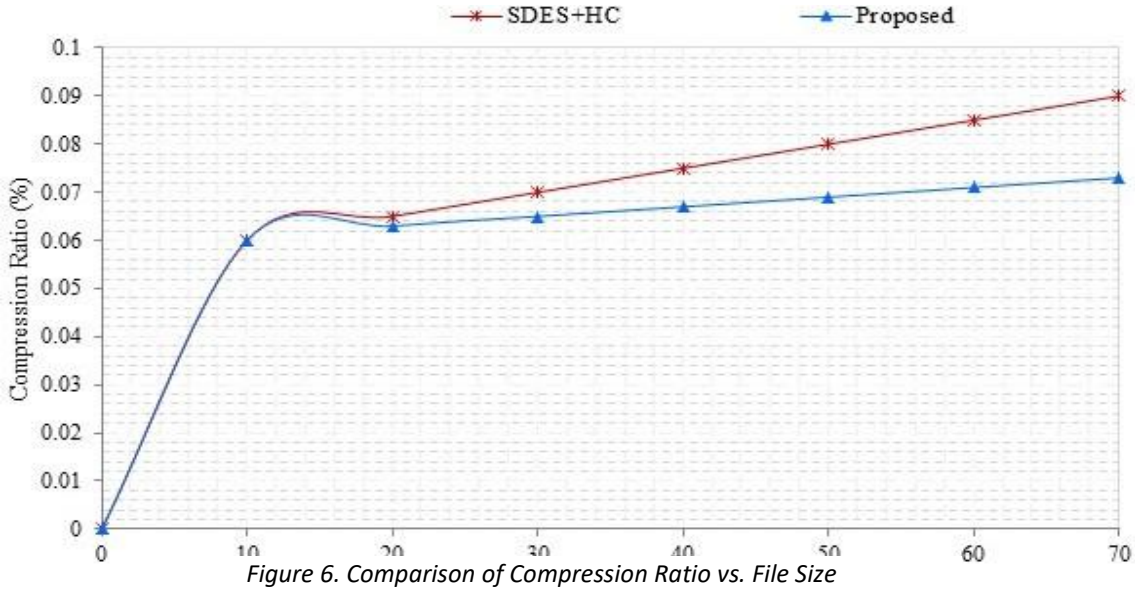

Figure 6. Comparison of Compression Ratio vs. File Size

ISSN: 2057-5688

alphabets. Our proposed LZMA compression technique can be solved all the problems of HC. In addition, LZMA is a fast compression technique so it can be easily suited for all applications. From the plots, we conclude that the average compression ratio for the proposed SADS-Cloud is 0.0585% and SDES with HC is 0.0656%.

5.2.3. Throughput: In general, each job assigned by the user must be completed within the time constraint. Throughput is a very important metric used for analyzing secure data transmission and storage over Cloud. It can be varied with respect to the processing speed of the designed system. To compute the performance of throughput, we considered two metrics such as clustering speed and data sharing speed to users. Fig 9 shows the comparison for throughput in terms of clustering speed. From the graph, we observed that the proposed clustering technique i.e. DBSCAN is providing better throughput performance than other clustering techniques at the time of clustering and data sharing to the CS. In SDES with HC, throughput is very less due to high processing overheads for individual files. Sharing speed is a data retrieval time for data users. It is computed by follows:

If any of the compression techniques show high CR, then it means that the technique is poor to minimize the original data. When compared to SDES with HC, our proposed SADS-Cloud scheme produces very less compression ratio. With the use of the LZMA compression technique, our proposed SADS-Cloud has provided a high compression ratio. In SDES with HC, the Huffman compression technique is proposed which ensemble changes according to frequencies and probabilities, but it does not consider Block of Symbols and it cannot be practical for large alphabets. Our proposed LZMA compression technique can be solved all the problems of HC. In addition, LZMA is a fast compression technique so it can be easily suited for all applications. From the plots, we conclude that the average compression ratio for the proposed SADS-Cloud is 0.0585% and SDES with HC is 0.0656%.

5.2.4. Throughput: In general, each job assigned by the user must be completed within the time constraint. Throughput is a very important metric used for analyzing secure data transmission and storage over Cloud. It can be varied with respect to the processing speed of the designed system. To compute the performance of throughput, we considered two metrics such as clustering speed and data sharing speed to users. Fig 9 shows the comparison for throughput in terms of clustering speed. From the graph, we observed that the proposed clustering technique i.e. DBSCAN is providing better throughput performance than other clustering techniques at the time of clustering and data sharing to the CS. In SDES with HC, throughput is very less due to high processing overheads for individual files. Sharing speed is a data retrieval time for data users. It is computed by follows:

$$Sharing\ Speed = \frac{Original\ File\ Size}{total\ execution\ time\ for\ retreival} \quad (2)$$

ISSN: 2057-5688

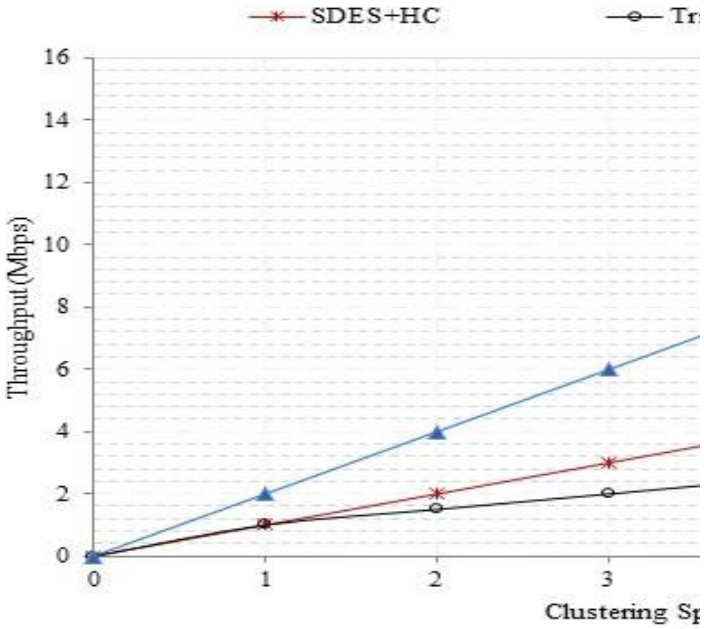

Figure 7. Comparison of Throughput vs. Clustering Speed

For optimal data retrieval, CS keep store encrypted data for data owners and managed by fractal index tree, which results in better performance in our proposed scheme. Data user requests data item to the CS. The CS retrieves data items from the nodes. With the lack of clustering, indexing and proper data management, previous scheme are very poor in throughput than our proposed scheme, which is can be seen in Fig.10.

ISSN: 2057-5688

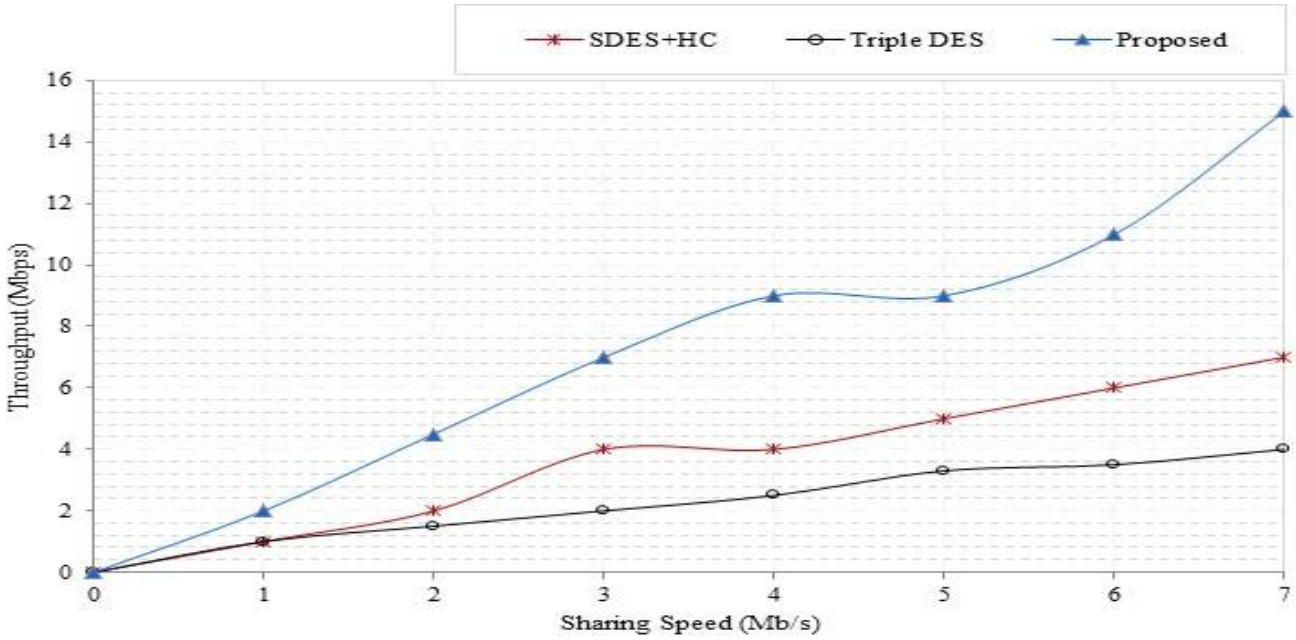

Figure 9. Comparison of Throughput vs. Sharing Speed

5.2.1. *Encryption Time:* Encryption time and decryption time are important metrics for any security technique. Based on the time performance, it is evaluated and proved. It is the time duration for performing data encryption (convert plain text into ciphertext). Fig 11 indicates the performance of the encryption time for the proposed SADS-Cloud with SDES with HC and Triple DES. We use a single secret key for encryption and decryption. SDES and Triple DES increases computational and communication overheads. For this reason, encryption time for the proposed scheme is very less than previous schemes. The average encryption time taken for the proposed scheme is 0.06875s, whereas SDES with HC requires 0.11s and Triple-DES requires 0.175s, respectively

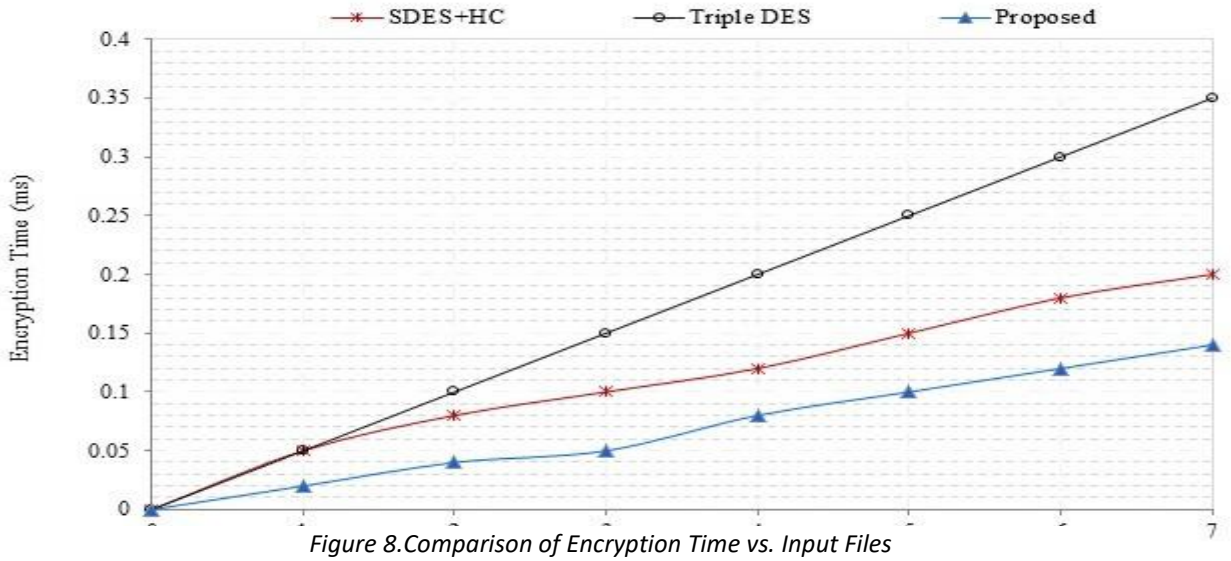

Figure 8.Comparison of Encryption Time vs. Input Files

ISSN: 2057-5688

5.2.2. *Decryption Time*: It is the time duration for performing data encryption (reverse of encryption). Decryption time for the proposed and previous schemes is depicted in Fig.12. From the plots, we observed that the proposed scheme has required a minimum time period for decryption operation

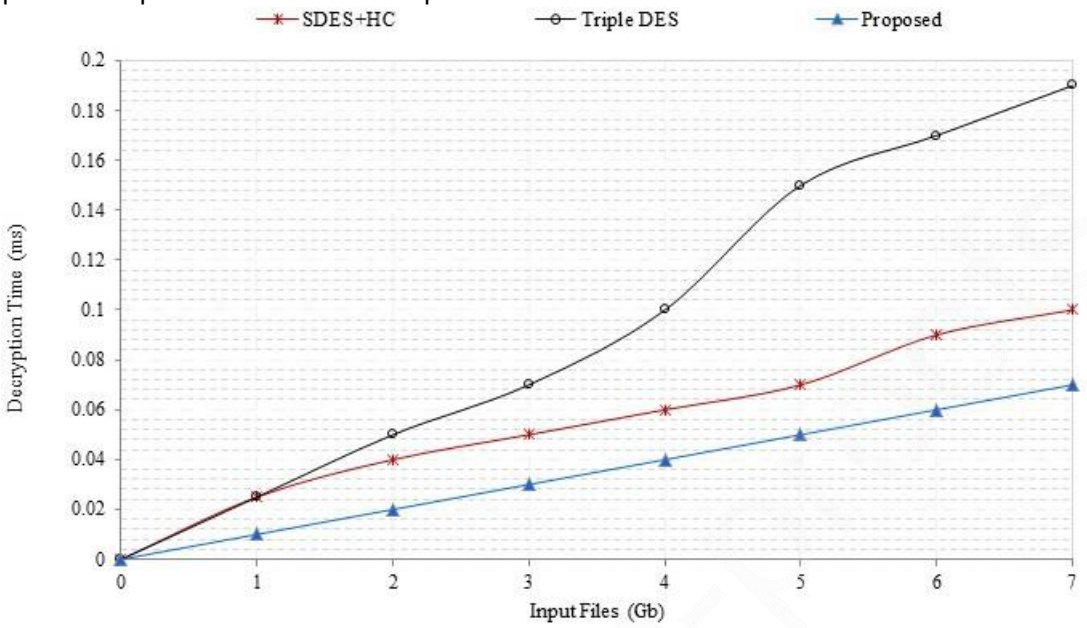

Figure 10. Comparison of Decryption Time vs. Input Files

5.2.3. *Efficiency (%):* In this paper, we considered efficiency is one of the metrics that illustrates the performance in terms of communication and computation overheads of the proposed algorithms with previous algorithms. From Fig 13 it can be observed that the proposed SADS-Cloud technique produces higher efficiency among the other previous security schemes. This describes the proposed scheme is powerful and it can be applied for any kind of data application. It protects the data from unauthorized access as well as reduces errors caused by security techniques.

ISSN: 2057-5688

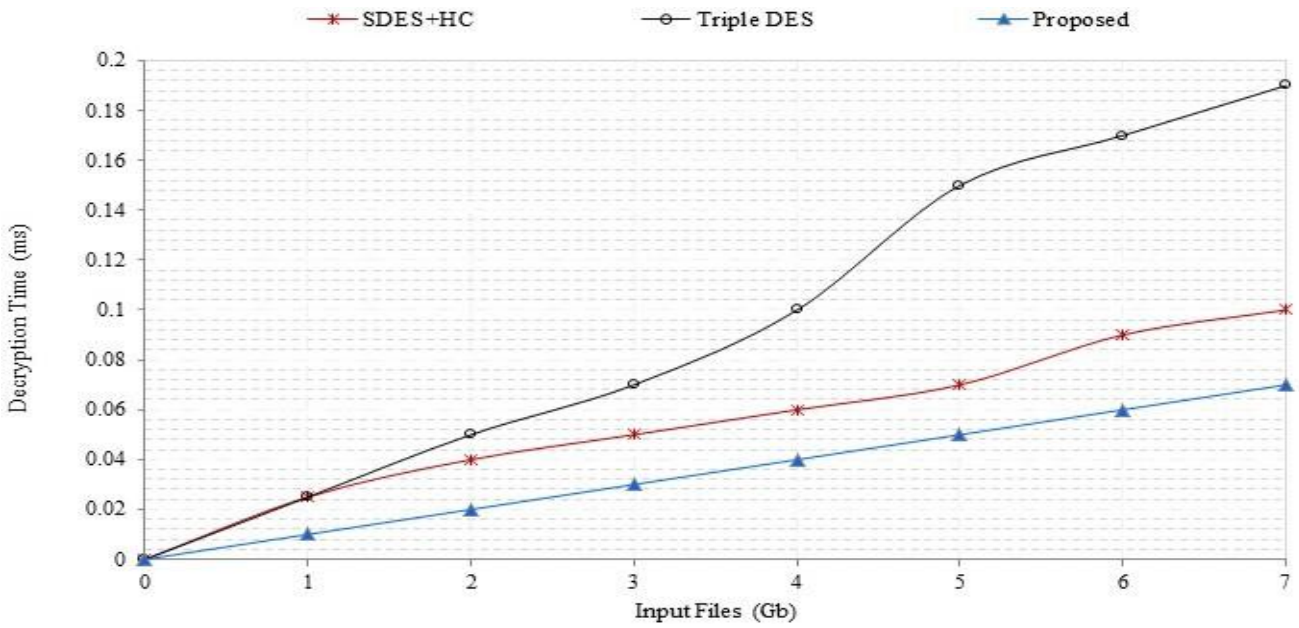

Figure 11. Comparison of Efficiency (%) vs. Input Files

ISSN: 2057-5688

1.2 Results Summary

In this section, we illustrate how the proposed scheme has provided better performance than previous approaches. Furthermore, we present the study of how the proposed scheme can be used in real-world application such as "E- Healthcare".

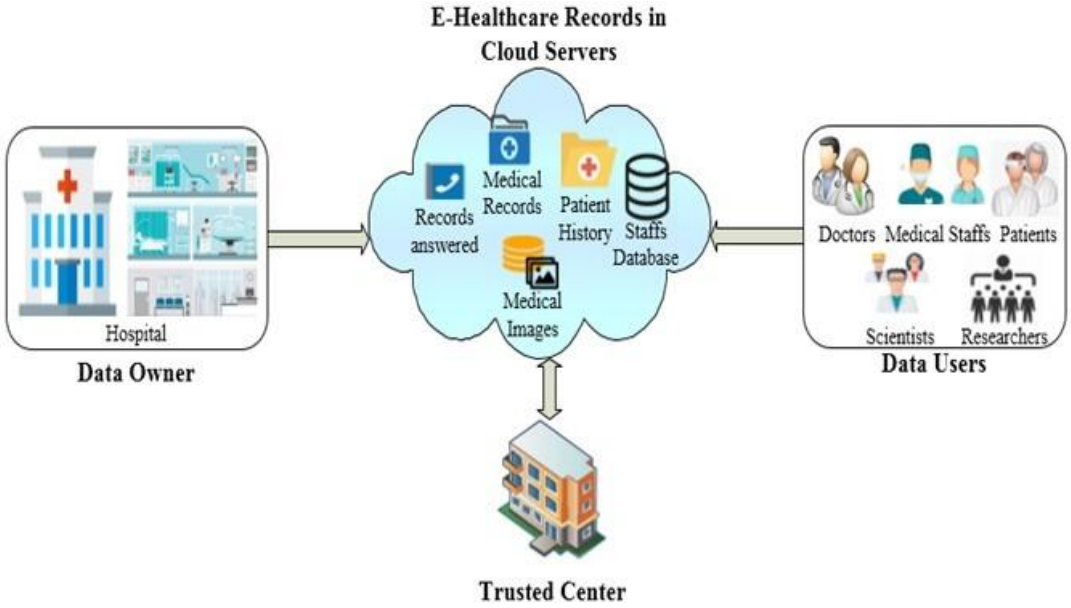

Figure 12: Use case (Healthcare Application)

Use Case (Healthcare Applications): The advent of big data has ushered in a plethora of new applications, including e-healthcare, smart homes, and manufacturing. For a Big Data-assisted Cloud environment, our proposed plan, SADS-Cloud, provides numerous benefits and applications. We go over some of the ways the proposed approach may be implemented in the E-healthcare software, which might have many different applications all around the globe. Everyone involved—doctors, patients, and others—can reap several advantages from e-healthcare applications. Medical professionals use a variety of devices to gather data regarding their patients' health. The data in this paper belonged to the hospitals, while the people who used it were the patients, physicians, nurses, researchers, and administrators. Chattaraj et al. (2018) and Masood et al. (2018) state that healthcare records make up CS, and that TC safeguards the privacy of data owners and users. Benefits of using big data in healthcare system development include simple integration for users and computer scientists, quick access to large amounts of stored data, and real-time data analysis. Many advantages accrue to patients who implement the proposed method, including remote access to their medical records, expedited assistance with treatment, security for their personal information, and the ability to see their own health statistics on a computer. An electronic healthcare program's use case diagram is shown in Figure 14.

A. Results Discussion: The goal is to make the process of storing large amounts of data in the cloud less vulnerable to security problems. For proof of how well the proposed combined method works, we compared it to some other security and grouping techniques in a number of different ways. In

ISSN: 2057-5688

addition, the suggested compression method gives better compression effectiveness to lower data and security costs. Numbers 7, 8, 9, 10, 11, 12, and 13 show that the suggested SADS-Cloud method works better than the old ones in terms of information loss, compression ratio, speed, encryption time, decoding time, and efficiency. These results were better than what was possible before. So, the results of the experiment showed that the suggested method is better and safer than all the others that are already out there. We got better results when we used the three big methods we suggested in our plan. As a result, we can say that our suggested plan meets all of our goals. Table 4 compares the proposed method to the old ones (SDES with HC and Triple DES) in terms of all performance measures. The security of our suggested work is checked against the most recent state-of-the-art studies, and the results are shown in Table 5. Lastly, we think that our plan for this growing and new study will help to show how it can be improved in the future.

Table.4 Comparison of the Proposed vs. Previous Approaches

Performance Metrics	Proposed vs. Previous Approaches		
	SDES with HC	Triple DES	Proposed
Information Loss	0.07%	0.175%	0.023%
Compression Ratio	0.0656%	-	0.0585%
Throughput	3.5Mbps	2.18Mbps	7Mbps
	3.625Mbps	2.225Mbps	7.18Mbps
Encryption Time	0.11s	0.175s	0.0687s
Decryption Time	0.054s	0.0943s	0.0325s
Efficiency	46.87s	35s	58.125s

Table. 5 Comparison of security properties with State of art papers.

Security Properties	**[**(R.S.Pippal, C.D.Jaidhar 2013)**]**	**[**(Yeh 2014)**]**	**[**(X. Li, Y.-P. Xiong, J. Ma 2012)**]**	**[**(T.Truong, M., T.Tran A, D.Duong 2017)**]**	**[**(Nafis and Biswas 2019)**]**	**[**(Ramya Devi and Vijaya Chamundeeswari 2018)**]**	**Ours**
Insider attack	√	*	√	√	√	√	√
Known-key attack	√	√	√	√	√	√	√
Replay attack	√	√	√	√	*	*	√
Perfect forward secrecy	√	*	√	√	*	√	√

ISSN: 2057-5688

Mutual authentication	√	√	*	√	*	√	√
Offline password guessing attack	*	*	*	*	*	*	√
User impersonation attack	*	*	*	*	*	√	√

ISSN: 2057-5688

Note. √ means that the scheme can provide the corresponding security property, and * indicates that it cannot achieve this.

2. Conclusion And Future Work

In this study, we talked about two important issues: user privacy and data protection in a Cloud setting that uses Big Data. This research talks about three ways to handle big data: managing it in the cloud, sharing it with users, and getting data owners to send it to other people. We suggested SHA-3 hashing as a safe way to authenticate users. This method would hash the user's message and store it in both TC and CS. The people who own the data send it to the cloud computer safely. The LZMA method is used to compress data so that the most of the cloud storing space that big data makes available is used. Next, we used SALSA20 Encryption MapReduce to encrypt the data. This makes the process of encrypting and decrypting the data much faster. The info is sent to CS once the encryption process is done. Before requests to get data can be handled, users must be verified. Once the file has been found, the secret keystream is used to read it. We looked at two ways to handle Big Data in the cloud: grouping with DBSCAN and indexing with Fractal Index Tree. The work that was recommended was done by our team on E-healthcare apps. We measured and compared things like speed, efficiency, information loss, compression ratio, and the time it takes to secure and decode data.

That way, encryption and decoding will be even faster. We plan to try this method with other encryption methods and work on more real-world uses in the future.

References

[1] Aditham, Santosh, and Nagarajan Ranganathan. 2018. "A System Architecture for the Detection of Insider Attacks in Big Data Systems." *IEEE Transactions on Dependable and Secure Computing* 15 (6): 974–87. https://doi.org/10.1109/TDSC.2017.2768533.

[2] Adnan, Nur Afifah Nadzirah, and Suriyani Ariffin. 2019. "Big Data Security in the Web-Based Cloud Storage System Using 3d-Aes Block Cipher Cryptography Algorithm." *Communications in Computer and Information Science* 937: 309–21. https://doi.org/10.1007/978-981-13-3441-2_24.

[3] Bearman, P. 2015. "Big Data and Historical Social Science." *Big Data & Society* 2 (2): 1–5. Bholat, D. 2015. "Big Data and Central Banks." *Bank of England Quarterly Bulletin* 55 (1): 86–93.

[4] Bronson, K., & Knezevic, I. 2016. "Big Data in Food and Agriculture." *Big Data & Society* 3 (1): 1–5.

[5] Castiglione, Arcangelo, Raffaele Pizzolante, Alfredo De Santis, Bruno Carpentieri, Aniello Castiglione, and Francesco Palmieri. 2015. "Cloud-Based Adaptive Compression and Secure Management Services for 3D Healthcare Data." *Future Generation Computer Systems* 43–44: 120–34. https://doi.org/10.1016/j.future.2014.07.001.

[6] Chattaraj, Durbadal, Monalisa Sarma, Ashok Kumar Das, Neeraj Kumar, Joel J.P.C. Rodrigues, and Youngho Park. 2018. "HEAP: An Efficient and Fault-Tolerant Authentication and Key Exchange Protocol for Hadoop-Assisted Big Data Platform." *IEEE Access* 6: 75342–82. https://doi.org/10.1109/ACCESS.2018.2883105.

ISSN: 2057-5688

[7] Fan, Kai, Shuyang Lou, Ruidan Su, Hui Li, and Yintang Yang. 2018. "Secure and Private Key Management Scheme in Big Data Networking." *Peer-to-Peer Networking and Applications* 11 (5): 992–99. https://doi.org/10.1007/s12083-017-0579-z.

[8] Gai, Keke, and Meikang Qiu. 2018. "Blend Arithmetic Operations on Tensor-Based Fully Homomorphic Encryption Over Real Numbers." *IEEE Transactions on Industrial Informatics* 14 (8): 3590–98. https://doi.org/10.1109/TII.2017.2780885.

[9] Gai, Keke, Meikang Qiu, and Hui Zhao. 2018. "Energy-Aware Task Assignment for Mobile Cyber-Enabled Applications in Heterogeneous Cloud Computing." *J. Parallel Distrib. Comput.* 111: 126–35. https://doi.org/10.1016/j.jpdc.2017.08.001.

[10] Gai, Keke, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. 2016. "Dynamic Energy-Aware Cloudlet-Based Mobile Cloud Computing Model for Green Computing." *Journal of Network and Computer Applications* 59: 46–54. https://doi.org/10.1016/j.jnca.2015.05.016.

[11] Goyal, Vikas, and Chander Kant. 2018. "An Effective Hybrid Encryption Algorithm for Ensuring Cloud Data Security." *Advances in Intelligent Systems and Computing* 654: 195–210. https://doi.org/10.1007/978-981-10-6620- 7_20.

[12] Hababeh, Ismail, Ammar Gharaibeh, Samer Nofal, and Issa Khalil. 2019. "An Integrated Methodology for Big Data Classification and Security for Improving Cloud Systems Data Mobility." *IEEE Access* 7 (c): 9153–63.https://doi.org/10.1109/ACCESS.2018.2890099.

[13] Hadj, Maryem Ait El, Mohammed Erradi, Ahmed Khoumsi, and Yahya Benkaouz. 2019. "Validation and Correction of Large Security Policies: A Clustering and Access Log Based Approach." *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, no. 1: 5330–32. https://doi.org/10.1109/BigData.2018.8622610.

[14] Hu, Chunqiang, Wei Li, Xiuzhen Cheng, Jiguo Yu, Shengling Wang, and Rongfang Bie. 2017. "A Secure and Verifiable Access Control Scheme for Big Data Storage in Clouds." *IEEE Transactions on Big Data* 4 (3): 341–55. https://doi.org/10.1109/tbdata.2016.2621106.

[15] Huang, Xinyi, Joseph K. Liu, Shaohua Tang, Yang Xiang, Kaitai Liang, Li Xu, and Jianying Zhou. 2015. "Cost-Effective Authentic and Anonymous Data Sharing with Forward Security." *IEEE Transactions on Computers* 64 (4): 971–83. https://doi.org/10.1109/TC.2014.2315619.