JOURNAL OF NORTHEASTERN UNIVERSITY

Volume 26 Issue 01, 2023 ISSN: 1005-3026 https://dbdxxb.cn/ Original Research Paper

A NEW EFFICIENT HYBRID CRYPTOGRAPHIC IMPLEMENTATION FOR INFORMATION SECURITY IN HADOOP

Kaparthi Ravi Kishore[1] and G. Shyama Chandra Prasad[1*]

[1]Research Scholar, Department of CSE, University College of Engineering (A), Osmania University, Hyderabad, Telangana- INDIA- 500007

[1*]Professor, Department of CSE, Matrusri Engineering College, Saidabad, Hyderabad, Telangana- INDIA- -500 059

ABSTRACT

An anticipated safe cloud computing system has been designed to address a significant challenge in creating a tailored Hadoop for the cloud. In this area, Hadoop was used to build and enhance the security of handling and collecting user data. Hadoop is one of the other Apache Big Data technologies, and it prepares massive volumes of data using the MapReduce architecture. One of the most important tools for addressing Big Data concerns is Hadoop. One of the most difficult challenges is securing data storage, and the Hadoop distributed file system (HDFS) lacks a clear security strategy. The suggested method uses public key cryptography to encrypt and protect each and every file stored in HDFS. Using the suggested data encryption technique, the collected data is encrypted in HDFS during the data collection process. In this paper, a hybrid encryption method for HDFS files is presented that combines two well-known asymmetric key cryptosystems (RSA and Paillier). Data is encrypted using the proposed cryptosystem before it is saved in HDFS. However, the RSA encryption algorithm usually requires a longer key to ensure data security. There are two methods for uploading files to the cloud: non-secure and safe. The hybrid system has a lower delay and higher computational complexity as compared to the RSA cryptosystem alone. Asymmetric encryption algorithms, such as RSA and Paillier, use public-key encryption and private key decryption, so it provides security in the cloud computing environment. The datasets required for the study is obtained from Amazon.

Keywords: cryptography, cloud computing and HDFS

1 INTRODUCTION

Many businesses employ cloud technology in the new digital era to store, analyse, and analyse petabytes of data related to their operations and clients [1] [2]. The present model for application delivery has been shaped by developments in cloud computing technology [3]. The benefits of adopting cloud computing are undeniable given its huge potential to offer simple, affordable access to powerful computer capacity. This trend is promoted by big data (BD) frameworks over cloud computing (BigCloud), which have the potential for greater scalability and elasticity than conventional solutions [5]. Data security is even more important because the outsourced data may contain private information, such bank records, proprietary research data, healthcare data, or government information. One of the top research areas for the cloud in the coming decade is data security and privacy [4]. The purpose of this essay is to keep cloud-based sustainable business operations operating effectively. The client's confidence in

Submitted: 30/01/2023 **Accepted**: 28/02/2023

data transit into and out of the cloud environment, as well as the storage and processing of sensitive data in an off-premises data centre, provide the key security problem. Although several essential aspects of the cloud (such multitenancy and virtualization) allow greater resource usage, they make it difficult to provide safe computation.

Large amounts of data are produced every second by rapid technological development. This enormous amount of data is processed by several applications, sensor networks, social networking sites, and small to large-scale businesses [6, 7]. Big data refers to any complicated, varied, and significant amount of data that cannot be processed by traditional data processing applications. Initially, big data was defined in terms of its volume, diversity, and velocity. Later, to provide equivalent descriptions of huge data, this has been expanded with a few additional attributes as authenticity, venue, validity, vocabulary, ambiguity, and value. However, while storing, analysing, and retrieving the desired results, these features frequently run into parallel problems. A significant volume of complicated data is analysed using big data analytics in order to find correlations and reveal hidden patterns. Big data security and extensive data use, however, are in conflict [8]. Big data collections can be processed and stored using the open-source Hadoop framework on a distributed, dependable, and scalable computer environment [9]. Large clusters or public cloud services frequently make use of Hadoop because of its low cost, speedy processing, fault tolerance, and flexibility. The Hadoop Distributed File System (HDFS) and MapReduce are the two primary parts of Hadoop [10].

On MapReduce, massive amounts of organised or unstructured data can be processed concurrently. To distribute massive amounts of data, HDFS makes use of many logical file systems on hard discs. Hadoop has weaknesses in its security features even if its processing capacity is significantly more than that of traditional data processing systems. Since the Hadoop design is not intended to be secure, it does not offer any security measures to safeguard data while it is being stored or transported. Many firms use big data for research and marketing, but they might not focus on security issues [11]. There would be significant reputational harm and legal implications as a result of the data breach. Big data from Hadoop manages delicate information including financial, personal, business, and confidential information like client, employee, and customer details. Aside from that, businesses store and analyse a lot of data, all of which needs to be protected in HDFS storage via encryption, threat detection, and logging methods. These processes assist the systems in early threat detection and data security for the users. Diversity is one of big data's key characteristics. Big data might be in the form of text, images, audio, video, and more. To safeguard data from the generating phase through the storage phase, numerous solutions have developed recently. Using data fabrication techniques and limiting access during data production improves data privacy. Data security and privacy are mostly addressed during the storage phase via encryption techniques [12]. The process of converting plain text into ciphertext is known as encryption. Data privacy is ensured by converting explicable data into inexplicable form, which enables users to authenticate properly [13] and blocks unauthorised users. The same encryption keys are used for both encryption and decryption. Encryption is still employed for effective data protection and confidentiality. Public

key algorithms and symmetric key algorithms are the two types of encryption algorithms. To secure massive data, existing research uses a variety of encryption algorithms, including AES, RSA, DES, and ABE [14, 15]. The two asymmetric encryption algorithms: RSA and ElGamal, and proposed a method to enhance the confidentiality of Hadoop data. This scheme complements the advantages of the two encryption algorithms, and improves the efficiency of data encryption and decryption. However, the RSA encryption algorithm usually requires a longer key to ensure data security. Kapil et al. [33] combined attribute encryption with a honey encryption algorithm to encrypt data stored in Hadoop.

The paper is organized as follows: Section 2 shows the related works, section 3 shows the proposed methodology, section 4 shows the simulation results and analysis and the paper is concluded in section 5.

II RELATED WORKS

Creating a secure Hadoop infrastructure is fundamentally a cloud computing challenge. The protection policy is applicable to a number of cloud services, including Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS). It also supports the majority of cloud computing requirements. This incident highlights the need for a strategy to manage these difficulties. Hadoop, which often uses the MapReduce design to organise massive volumes of data from the cloud system, may be a strategy used to combat this big data issue. The Hadoop Distributed File System does not have any policies in place to guarantee the security and privacy of the files stored there (HDFS). Sensitive data security on the cloud could be a major issue, one where encryption techniques are crucial. To cypher the data maintained in HDFS, Yousif et al. [16] offer a hybrid technique combining two well-known asymmetric key cryptosystems (RSA and Rabin). Therefore, the proposed cryptosystem is used to cypher the data before it is stored in HDFS. The user of the cloud could upload files in two different ways under the suggested system: securely or insecurely.

AES and OTP algorithms integrated on Hadoop were utilised in this work by Mahmoud et al. [17] to increase the performance of file encryption and decryption. In the HDFS, files are encrypted, and the Map Task decrypts them. In earlier research, encryption and decryption were performed using the AES technique, which resulted in a 50% increase in the size of the encrypted file. As the encrypted file grew 20% larger than the original file, the suggested method boosted this ratio. The efficient authentication protocol for Hadoop is a fault-tolerant authentication technique that Chattaraj et al [18] propose for the Hadoop architecture (HEAP). The operating system-based authentication, password-based approach, and delegated token-based techniques, which are currently used in Hadoop, are the three state-of-the-art authentication mechanisms that HEAP addresses. The two-server-based authentication mechanism used by HEAP. HEAP uses elliptic curve cryptography and an advanced encryption standard to generate digital signatures that are used to authenticate the principal. To maintain platform security, Yang and Min [19] primarily focus on implementing access control on users. In order to implement front-end authorization, which can accurately reflect and manage the flexible nature of the complex access control process in the Hadoop platform, we first use access proxy integrated with the attribute-based access control (ABAC) model. By using access

proxy, we can also release back-end resources from the complex authorization process. Additionally, the access proxy maintains a list made up of trust threshold values provided by each resource in accordance with its significance in order to guarantee the fine-granularity of authorization.

The user's trust evaluation value is obtained by the access proxy through communication with the blockchain network and is a crucial component of the dynamic permission decision process. The blockchain network specifically operates in both on-chain and off-chain modes. The associated hash value is anchored on-chain, and the user's past behaviour data is kept off-chain. As a result, the user's trust value is assessed using his previously recorded blockchain platform activity. In the meantime, it is possible to ensure the validity of user behaviour data, hence ensuring the accuracy of trust evaluation results. Based on the ideas of a role-based access control system, Shetty et al. [20] suggested a multi-layer policy-based access control strategy for the Hadoop environment. The suggested paradigm applies access control for data owners and offers a mechanism to prevent unwanted access to cluster resources. Asymmetric encryption algorithms, such as RSA and ECC [12], use public-key encryption and private key decryption, so it provides security in the cloud computing environment. Studies have shown that 160-bit ECC encryption is equivalent to 1024-bit RSA encryption, and 210-bit ECC encryption is equivalent to 2048-bit RSA encryption [13]. Therefore, for the security issues of data blocks in Hadoop, we propose to use homomorphic encryption and ECC lightweight encryption for data that needs to be calculated and ordinary data, respectively. This solution can ensure data security while considering encryption efficiency.

III PROPOSED METHODOLOGY

The suggested method uses public key cryptography to encrypt and protect each and every file stored in HDFS. Using the suggested data encryption technique, the collected data is encrypted in HDFS during the data collection process. In this paper, a hybrid encryption method for HDFS files is presented that combines two well-known asymmetric key cryptosystems (RSA and Paillier). Data is encrypted using the proposed cryptosystem before it is saved in HDFS. However, the RSA encryption algorithm usually requires a longer key to ensure data security. There are two methods for uploading files to the cloud: non-secure and safe. The hybrid system has a lower delay and higher computational complexity as compared to the RSA cryptosystem alone. Asymmetric encryption algorithms, such as RSA and Paillier, use public-key encryption and private key decryption, so it provides security in the cloud computing environment.

3.1 RSA AND PAILLIER ALGORITHM IN HADOOP DISTRIBUTED FILE SYSTEM

Since Hadoop is the top provider of services for large-scale cloud computing and warehouses, it must use cutting-edge encryption methods to ensure security. The plan of attack is to stack two public-key cryptosystems on top of one another. The idea of hybridization was created to extend security while overcoming the drawbacks of using a particular cryptosystem alone. Each record related to HDFS is expected to be encrypted before being entered into the database.

Algorithm 1: - Key generation procedure

Input: prime number of size n bits.
Output: The N=p_q public key is returned.

The message was sent between user A and user B.

1. Create two primes that are roughly the same size and satisfy the condition (p and q are 3 mod 4;$2np2n + 1$) to solve this problem.

2. Calculate N, which is the product of the multiplications of p and q ($N = p * q$). Although the clear text or message (m) is raised to power 2e rather than e, as is the case with the Rabin cryptosystem, the encryption process is still essentially based on the Rabin encryption approach.

3. The cipher text c is then calculated as follows:

$$C = M^2 \ mod \ N \quad (1)$$

The model for generating keys, both public and private, is based on the RSA process. The degree of security offered by the RSA algorithm may be impacted by the capacity to factor huge integers. An illustration of the cryptographic system is Algorithm1. The following sections provide a detailed explanation of the encryption procedures: The HDFS client is eligible for significant generational enhancements (public and personal keys). The file is encrypted using the suggested hybrid technique, and it is also buffered to HDFS to mimic an unstructured file. The first step in the encryption procedure is when the HDFS starts sending the cypher files to the information nodes. The High Density File System (HDFS) is made up of a reputation Node, which supervises client access to particular encryption files, saves Metadata, manages the namespace of the filing system, and stores metadata. One or more segments that have been gathered in a massive collection of knowledge nodes are used to produce the cyphering data.

3.2 THE PAILLIER HOMOMORPHIC ENCRYPTION

Typically, text files in HDFS are used to contain the data that needs to be calculated (.txt). We encrypt the data using the Paillier homomorphic encryption algorithm to prevent leakage when the parallel computing framework is applied to huge data computing workloads. Following are the steps:

Step 1 Algorithm initialization:
(1) Randomly generate two large prime numbers p and q, and let $n = p \times q$;
(2) Take the least common multiple λ of $p-1$ and $q-1$, which is specifically expressed as formula 1:

$$\lambda = lcm\,(p-1)\,(q-1) \quad (1)$$

Where *lcm* represents the least common multiple.

(3) Choosing a random integer g needs to satisfy the order that n can divide g, which is specifically expressed as formula 5:

$$gcd\,(g\hat{}\lambda \bmod n\hat{}2, n) = 1 \quad (2)$$

where *gcd* represents the greatest common divisor, the public key is $pk = (n, g)$, and the private key is $sk = (p, q)$;

Since text and character strings make up the majority of the data to be encrypted, we employ a buffered character stream to read in plaintext and write out cipher text in order to speed up reading and writing files during encryption and decryption. The buffered character stream accelerates data processing by minimizing I/O (Input/Output) operations required while reading and writing data. The conventional physical stream can only read/write 1 byte of data, whereas the buffered character stream can read/write 8k (the default number) bytes of data in a single I/O transaction. In other words, for the same amount of data, employing buffered character streams can dramatically cut down on the number of I/O operations. Additionally, there are specific layout criteria when utilizing the MapReduce distributed computing framework to calculate data, and the text data is typically a row of data. We have updated the Paillier method to make sure that MapReduce can typically recognize the layout rules after encrypted storage. When reading data, we use string segmentation and independently encrypt the string for each field. We can format the cipher text when writing it such that it looks like plaintext.

Algorithm2: -Paillier homomorphic encryption

INPUT: N=$p * q$.

OUTPUT: The encoded cipher-text C.

User A sends a message to user B, who then receives it. User B is required to complete the following steps to encode the data:-

(a) Find out what A's true public key is (N=p_q).

(b) Explain how information such as the number m in the period should be interpreted.

(c) Determine the value of C = (M_2) module N.

(d) Copy the encoding text to the variable A.

The decoding procedure mimics the Rabin decryption scenario, in which the cypher text is raised to the power of the private key (d). The output, however, is the message raised to

power 2 rather than the direct message (M). To get the message, M first solves V using the Chinese Remainder Theorem (CRT) and the pair (p,q). M then returns four messages, each referred to as M1, M2, M3, and M4. The following step is to loop through all of the messages and compute

$$(W_i = \frac{C - M_i^2}{N}) \quad (2)$$

The following process is discussed in more detail in Algorithm 3:

Algorithm3: Process for decrypting the suggested algorithm

INPUT: Both the private key and the encoded text were accepted (d, p, q).
OUTPUT: Use the M character encoding to translate the encoded content into plain text

(a) Use the formula V=C mod d to find V=C mod.

(b)Utilizing the CRT, determine the square root of V. (p, q)

(c) The four potential signals are M1, M2, M3, and M4. Pay attention to all four.

(d) Calculate ($W_i = \frac{C - M_i^2}{N}$), in a loop.

(e) Return the simple text Mi that generates the outcomeW_i.

Below is a mathematical illustration of the Rabin RZ:

The scenario is that A (Bob) sends B (Bob) his public key, and B (Bob) encrypts to Along. Along will calculate N=707968400363899 and d=7032635671 using the primes p=100669 and q=69859 as well. Let's imagine Bob wants to send Alice a message with the ID M=1439948310. Bob will calculate both ways. Let's imagine Bob wants to communicate with Alice. Bob will calculate and send to Alice the result of 519659206359828=14399483102 (mod 70796840036899). Along computes the decryption key. Then, Alice computes the four square roots of 3691358296 modulo d: M1=3890433108, M2=1439948310, M3=5592687361, and M4=3142202563 using the CRT and his private keys. the right message can then be determined. In computing for i=1 to 4, along: Wi=C-Mi2/N .

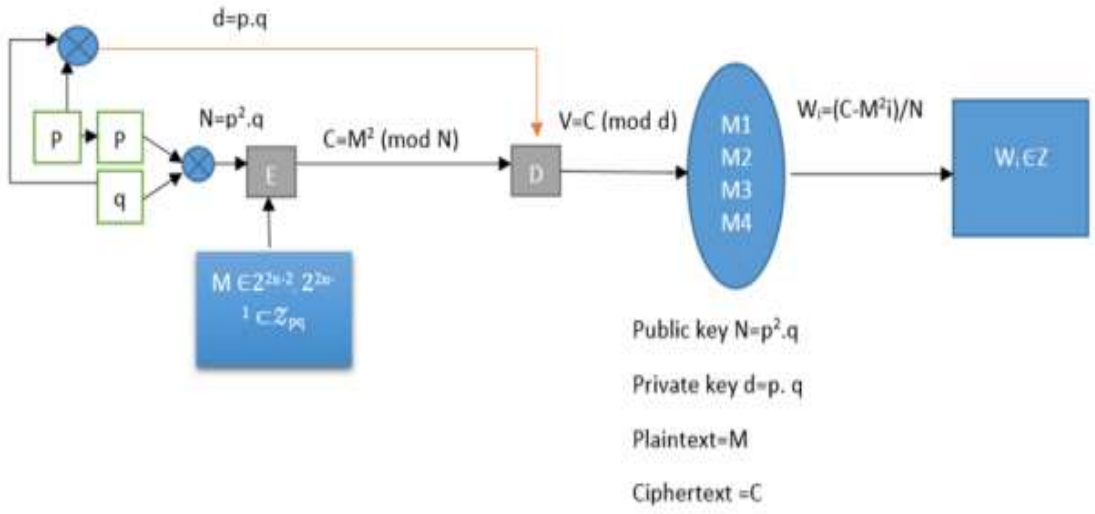

Fig 1: Flowchart of the data encryption procedure

In this illustration, just M2 generates WZ. The data nodes effectively segregate production, replication, and deletion based on instructions from the name node by using the proposed approach coding scheme. Additionally, data regarding the client is gathered from many sources and scrambled using asymmetric mechanism cryptographic tools on the server. A short time after encoding, data is saved in the cloud, notably using the Hadoop File System (HDFS), where a network of computers can access it. The server will deliver the encrypted data whenever a user requests it so that it can be decoded. The user then employs the associated keys to extract the decrypted data, as suggested in this research, by using a hybrid approach. A flowchart of the data encryption procedure, which entails numerous steps, is shown in Figure 1.

IV SIMULATION RESULTS AND ANALYSIS

The MapReduce and HDFS frameworks are used to gauge how encrypted HDFS performs. Each node has a 1 TB hard drive, 16 GB of RAM, and an i5 processor. The input file, which resides in the cluster on a distributed file system, is divided into groups of equal size to facilitate and simplify the enormous amounts of data processing in parallel on large clusters of hardware. This enables processing enormous amounts of data in parallel on large clusters of hardware in a suitable and nearly error-free manner.

Table 1: The computational complexity of various methods

Method	Encryption	Decryption
RSA	T(c)= O $(\log n)^3$	T(M)= O $(\log n)^3$
Pailier	T(c)= O $(5\,n^2 + 2n)$	T(M)= O $(2\,n^3 + (12\,n^2 + 2\,n)^3$

Proposed	T(c)= O $(4n^2 + 3n)$	T(M)= O $(2n^3 + (12n^2 + 4n)^3$

In Table 1, the computational complexity of the proposed technique is contrasted with that of the RSA and Paillier cryptosystems. It is evident that the computational complexity of the suggested method is lower than that of the individual cryptosystems (RSA or Rabin

Time required to decrypt the Hadoop separation dataset files and change them to plaintext using the private key is measured in minutes and seconds, whether using the RAS alone or the suggested approach. Seconds are used to measure it. In this instance, the Encryption Time essentially equals the system's present time prior to encryption, letting the system's present time following encryption, resulting in the encryption time being given. As illustrated in the example below, the Decryption Time is equivalent to the System Current Time before Decryption minus the System Current Time after Decryption.

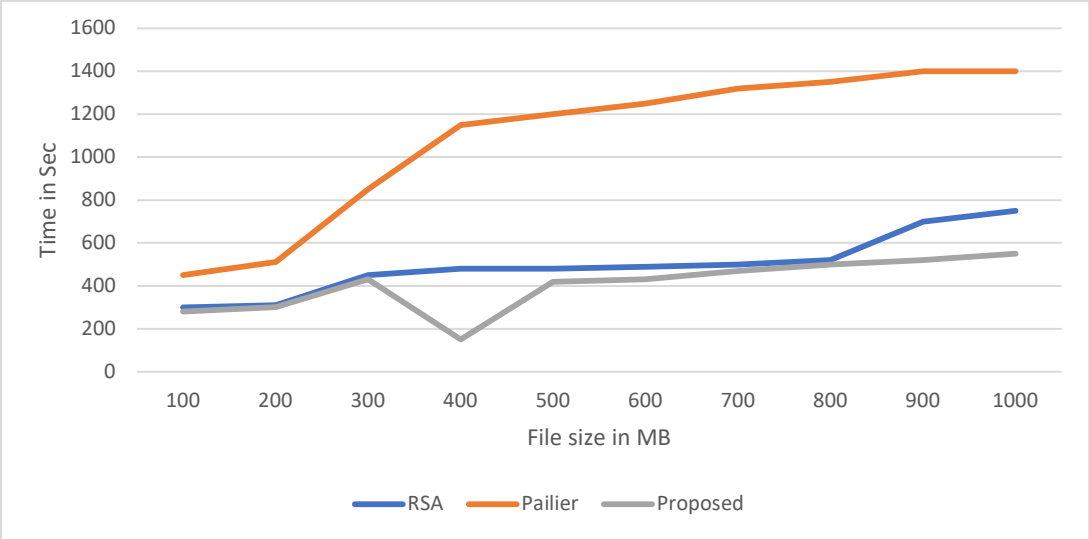

Figure 2: Comparison of encryption time

The effects of linking encryption techniques like those employed by RSA and Paillier with the suggested method were shown in Figure 2 using various file sizes. It is evident that the proposed technique displayed efficient time consumption when compared to the RSA and Paillier methods when file sizes ranging from 100 MB to 1 GB with a step size of 100 MB or higher are considered. Furthermore, compared to the standard RSA and Paillier methods, the suggested solution for the encryption stage is much quicker.

Figure 3: Comparison of decryption time

As illustrated in Figure 3 the suggested technique, RSA, and Paillier all have a time window throughout the decryption phase. Several various sizes of encrypted files have been applied to the present stage. When all of the previously mentioned systems (including the proposed technique) are applied, the suggested ciphered approach has a shorter decryption time than the other ways.).

V CONCLUSION

The suggested Hybrid cypher asymmetric key approach encrypts the file's content before storing it in HDFS by shielding it from various network incursions, as shown in the following figure. Since the files have already been encrypted before being collected in Hadoop, it is now possible to save files or data there without worrying about security concerns. The proposed system contains the following extra elements in addition to the most well-known cloud computing service models, such as Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service: (PaaS). It aids in data transfer authentication as well as information management and security issues (Verification, Credibility, Ease of Access, and Secrecy) in protection. In comparison to the standard RSA and Paillier methods, the proposed encryption approach is substantially faster. The suggested method, RSA, and Paillier all have periods throughout the decryption phase. The proposed ciphered method is faster to decode than the other methods when all of the systems previously mentioned (including the suggested method) are used. The proposed method demonstrated exceptional performance when used to a wide range of file sizes during the encryption and decryption stages, despite its additional complexity (double the computational complexity in the decryption stages). Longer-term work would be required for the incorporation of the RSA asymmetric key cryptosystem. It was found that the encryption and decryption rates for encryption and decryption were noticeably greater than the average, continuing the effects of the complex system investigation. The suggested solution, which depends on quick blocks for encryption and decryption, has the potential to generate an ideal concept for the purpose. The Rabin cryptosystem's decryption has been broken down more effectively than other systems have in the past.

References

[1] I. A. T. Hashem et al., "The rise of "big data" on cloud computing: Review and open research issues," Inf. Syst., vol. 47, pp. 98–115, Jan. 2015.
[2] F. M. Awaysheh, F. Toms Pena, and J. C. Cabaleiro, "EME: An automated, elastic and efficient prototype for provisioning hadoop clusters on-demand," in Proc. 7th Int. Conf. Cloud Comput. Serv. Sci., 2017, pp. 737–742.
[3] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," Future Gener. Comput. Syst., vol. 79, pp. 849– 861, 2018.
[4] R. Buyya et al., "A manifesto for future generation cloud computing: Research directions for the next decade," ACM Comput. Surv., vol. 51, no. 5, pp. 1–38, 2018.
[5] F. M. Awaysheh, M. Alazab, M. Gupta, T. F. Pena, J. C. Cabaleiro, "Nextgeneration big data federation access control: A reference model," Future Gener. Comput. Syst., vol. 108, pp. 726–741, Jul. 2020
[6] Q. Hou, M. Han, Z. Cai, Survey on data analysis in social media: A practical application aspect, Big Data Mining and Analytics, 3(4): 259–279, 2020
[7] A. Banik, Z. Shamsi, D.S. Laiphrakpam, An encryption scheme for securing multiple medical images, Journal of Information Security and Applications, 49: 1–8, 2019, doi: 10.1016/j.jisa.2019.102398.
[8] T. Wang, Z. Zheng, M.H. Rehmani, S. Yao, Z. Huo, Privacy preservation in big data from the communication perspective – A survey, IEEE Communications Surveys & Tutorials, 21(1): 753–778, 2019, doi: 10.1109/COMST.2018.2865107.
[9] X. Wang, M. Veeraraghavan, H. Shen, Evaluation study of a proposed Hadoop for data center networks incorporating optical circuit switches, IEEE/OSA Journal of Optical Communications and Networking, 10(8): C50–C63, 2018, doi: 10.1364/JOCN.10.000C50.
[10] J. George, C.-A. Chen, R. Stoleru, G. Xie, Hadoop MapReduce for mobile clouds, IEEE Transactions on Cloud Computing, 7(1): 224–236, 2019, doi: 10.1109/TCC. 2016.2603474.
[11] G.S. Bhathal, A. Singh, Big Data: Hadoop framework vulnerabilities, security issues and attacks, Array, 1–2: 1–8, 2019, doi: 10.1016/j.array.2019.100002.
[12] R.R. Parmar, S. Roy, D. Bhattacharyya, S.K. Bandyopadhyay, T.-H. Ki, Large-scale encryption in the Hadoop environment: challenges and solutions, IEEE Access, 5: 7156–7163, 2017, doi: 10.1109/ACCESS.2017.2700228.
[13] J. Samuel Manoharan, A novel user layer cloud security model based on chaotic Arnold transformation using fingerprint biometric traits, Journal of Innovative Image Processing (JIIP), 3(01): 36–51, 2021, doi: 10.36548/jiip.2021.1.004.
[14] H.-Y. Tran, J. Hu, Privacy-preserving big data analytics a comprehensive survey, Journal of Parallel and Distributed Computing, 134: 207–218, 2019, doi: 10.1016/j.jpdc.2019. 08.007.
[15] N. Eltayieb, R. Elhabob, F. Li, An efficient attribute-based online/offline searchable encryption and its application in cloud-based reliable smart grid, Journal of Systems Architecture, 98: 165–172, 2019, doi: 10.1016/j.sysarc.2019.07.005.
[16] Yousif, Raghad Z., Shahab W. Kareem, and Shadan M. Abdalwahid. "Enhancing Approach for Information Security in Hadoop." *Polytechnic Journal* 10, no. 1 (2020): 81-87.

[17] Mahmoud, Hadeer, Abdelfatah Hegazy, and Mohamed H. Khafagy. "An approach for big data security based on Hadoop distributed file system." In *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pp. 109-114. IEEE, 2018.
[18] Chattaraj, Durbadal, Monalisa Sarma, Ashok Kumar Das, Neeraj Kumar, Joel JPC Rodrigues, and Youngho Park. "HEAP: an efficient and fault-tolerant authentication and key exchange protocol for Hadoop-assisted big data platform." *IEEE Access* 6 (2018): 75342-75382.
[19] Yang, Min. "TDACS: an ABAC and Trust-based Dynamic Access Control Scheme in Hadoop." *arXiv preprint arXiv:2011.07895* (2020).
[20] Shetty, Madhvaraj M., D. H. Manjaiah, and Ezz El-Din Hemdan. "Policy-Based access control scheme for securing hadoop ecosystem." In *Data Management, Analytics and Innovation*, pp. 167-176. Springer, Singapore, 2019.