Hadoop Distributed File System Hybrid Encryption Algorithm for Big Data Security

[1]Khushi Ram, [2]Prof (Dr.) Mukesh Singla

Department of Computer Science and Applications
Baba Mastnath University Rohtak

Abstract: Big Data has become a crucial asset for organizations, but its security remains a significant concern. Traditional encryption algorithms may not provide sufficient protection for large-scale distributed systems like Hadoop Distributed File System (HDFS). To address this challenge, this paper present a hybrid encryption algorithm that combines the strengths of both symmetric and attribute-based encryption techniques for enhanced security of Big Data in HDFS.The present algorithm utilizes the Advanced Encryption Standard (AES) as a symmetric encryption algorithm and Cipher Policy Attribute-Based Encryption (CP-ABE) for attribute-based encryption. This two-tier security approach enables the encryption of file attributes using CP-ABE and the encryption of the file itself using AES with the key generated by CP-ABE. This combination enhances data security while also facilitating the identification of intruders during the decryption process.The evaluation of the present hybrid encryption algorithm is performed by comparing it with traditional encryption algorithms, including Data Encryption Standard (DES), Triple DES (3DES), and Blowfish. The results demonstrate that the hybrid model outperforms these conventional algorithms in terms of encryption time, decryption time, throughput, and efficiency.

Keywords: Big data security, Hadoop, data encryption and decryption.

I. INTRODUCTION

With the growth of big data, the need for secure and efficient data storage and transmission has become increasingly important. Traditional encryption algorithms, such as AES and RSA, provide good security for small data, but are not optimized for big data due to their high computational overhead and limited scalability. This has led to the development of new encryption algorithms that are specifically designed for big data security.The characteristics of big data, including volume, velocity, variety, veracity, venue, validity, vocabulary, vagueness, and value, make it challenging to store, process, and analyze using traditional methods. Big data analytics aims to extract insights and knowledge from these large and complex datasets, but this process is often hindered by security concerns, such as data breaches and unauthorized access.Hybrid encryption algorithms are a promising solution for big data security, as they combine the strengths of symmetric and asymmetric encryption to achieve both security and efficiency. In a hybrid encryption scheme, a symmetric encryption algorithm is used to encrypt the data, and the key used for the symmetric encryption is itself encrypted using an asymmetric encryption algorithm, such as RSA or Elliptic Curve Cryptography (ECC).
Hadoop is an open-source software framework that provides a distributed computing platform for processing and storing large datasets across clusters of computers. It is designed to be scalable, reliable, and fault-tolerant, and it can handle a wide variety of data types and formats. Hadoop is widely used in industry and academia for big data processing and analysis, and it includes several key components, such as the Hadoop Distributed File System (HDFS) and the MapReduce programming model.
This research paper aims to provide a comprehensive review of the literature on hybrid encryption algorithms for big data security, including their design, implementation, and evaluation. The paper will discuss the advantages and limitations of different schemes proposed in the literature, and identify the key research challenges and opportunities for future work. By providing a critical analysis of the state-of-the-art in this field, this paper aims to contribute to the development of more secure and efficient solutions for big data security.

II. RELATED WORK

Big data security and encryption methods are briefly addressed in this section of the literature review. The security and privacy features of big data applications are constantly being improved by researchers employing various encryption techniques [1].A privacy-preserving auction technique is used in the homomorphic cryptography and secure network protocol design described in [2] to increase data confidentiality and build trust between the user and third-party service provider.The completely homomorphic encryption scheme described in [3] prevents risks to data privacy from both inside and outside the big data environment. The robust cryptosystem examines the assignments and divides the computation and data into two distinct subsets. The system's capacity to process data is greatly sped up and accuracy is increased through this method.A method for complicated data encryption and computing that accelerates processing is published in and is said to be able to handle a variety of situations. A safe data storage system that uses adaptive crypto acceleration and dynamically optimises the big data file operating modes is proposed for heavy data encryption. The research achieves a superior trade-off in comparison to previous software and hardware accelerators.The data analysis methodology described in [4] effectively addresses privacy concerns in the exchange of health information by using an encryption technique. Patient-centric data access control mode addresses the concerns about privacy in medical records and the need for data encryption. The patient file is encrypted via the proposed RSA-based encryption, which communicates using multiple domains.The findings in [5] highlight the challenges the user encounters when the data is outsourced. The main goals of the research project are data privacy and secrecy, and multi-keyword searchable encryption helps to achieve these goals. The formation of the probabilistic trapdoors enhances data security and resistance to attacks. Data secrecy in huge data streams is guaranteed by the selective encryption technique described in [6]. Data integrity and confidentiality are security factors that affect how reliable the obtained data is. To improve data stream encryption and decryption performance while maintaining data integrity and confidentiality, the authors employ the selective encryption method.The limitations of traditional cryptography methods were highlighted in attribute-based encryption, which was presented in [7]. In a big data environment, the encryption model addresses the significance of fine-grained access control. A vast volume of data may be handled more easily thanks to the flexible policies that improve access control on encrypted data. Cryptographic acceleration is used to improve performance and is used to evaluate the performances based on 10 different factors. The main advantages of this encryption model are that it requires very little memory and power. The drawback of traditional attribute-based encryption (ABE) algorithms is removed by the hybrid attribute-based encryption (ABE) model proposed in [8]. The described hybrid model adds proxy re-encryption to change ABE's ciphertext into identity-based encryption (IDE) ciphertext since the policies set by the traditional models become obsolete after a certain amount of time.

The CP-ABE technique was used by the authors of [9] to examine the problems with unauthorised data access and privacy issues in huge data clouds. The encryption algorithm solved the requirement of multi-authority access control and data privacy. The role hierarchy algorithm and hierarchy access structure offered user data and fine-grained access. The hierarchy algorithm's two main advantages are computation speed and minimal storage usage.

Big data security analysis should take attack and intrusion detection into account because they have an impact on the privacy and confidentiality of the data. For the purpose of enhancing data security, numerous attack detection models and intrusion detection approaches have emerged. A two-step attack detection technique described in [10] secures communication protocol by means of instruction sequences. To further analyse the needs of the system, these instruction sequences are matched with the nodes. The encryption and decryption process described in [11] takes into account the shortcomings of attribute-based encryption pairing processes, which have a negative impact on the encryption system's performance. A lightweight, fine-grain data sharing methodology is used for outsourced data operations to get around this problem, improving overall data security and preventing decryption key exposure.

According to the literature, key management and authentication processes for complicated data management are highly challenging for cryptographic techniques.Homomorphic encryption offers improved data protection, however the encryption method is exceedingly slow and ineffective. In contrast, fully homomorphic encryption offers higher data privacy and usability. The technique is not practical for huge data applications due to concerns with accuracy or sluggish calculation time. Although RSA-based encryption offers greater authenticity and confidentiality, it is slow for processing large amounts of data.When encrypting common data, traditional encryption algorithms work effectively. The computation times for the traditional encryption algorithms, however, rise for big data due to the volume and diversity of the data. A better system can be achieved by considering hybrid encryption models, which improves encryption performance.This conclusion leads to the conclusion that hybrid encryption solutions can enhance the security of big data. In light of these findings, our research suggests a hybrid encryption strategy for big data security in a distributed file system environment that aims to provide optimum speed with minimal computation cost.

III. HYBRID ENCRYPTION ALGORITHM

To protect the massive data stored in HDFS, CP-ABE and AES are combined in the proposed hybrid encryption technique. In comparison to conventional encryption methods, the double encryption process offers higher data protection. ABE has recently drawn increasing attention because of its decentralised access control and secure communication capabilities under changing circumstances. A user cannot, however, create any policies or processes that would specify the encryption process. Access control policies in the form of cypher text policies are established in order to provide users more control over how the encryption process is accessed.

Users can specify the encryption and decryption process's policy in addition to attributes. The data in transport and storage is additionally encrypted-secured by these access control settings. By combining the concepts of private and public keys into a single attribute-based idea, ABE effectively handles user requirements. Based on the logical combinations of the attributes, the attribute policies stated in ABE are obtained. The qualities and their logical combinations can be framed in an ABE according to the application and user demands. Static and dynamic properties are both supported by the proposed hybrid encryption system (Figure 1). In CP-ABE, only the attributes are encrypted; the full block is not encrypted. Secret or symmetric cyphers use AES to generate a 128-bit key, which is then used for both encryption and decryption. The ABE algorithm is first run as part of the two-step procedure, and then AES is used. If the algorithms are used alone instead of in combination, CP-ABE relies on some random user private key, which is needed to be obtained based on attributes.

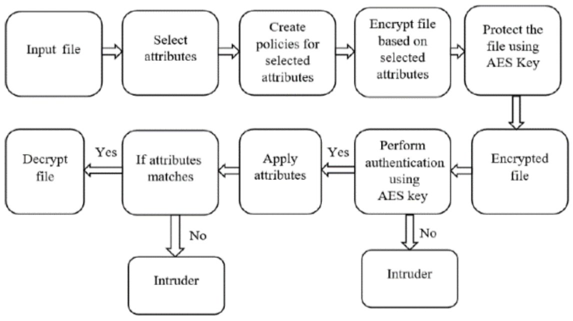

Input file
Select attributes
Create policies for selected attributes
Encrypt file based on selected attributes
Protect the file using AES Key
Encrypted file
Perform authentication using AES key
Yes
Apply attributes
If attributes matches
Yes
Decrypt file
No
Intruder
No
Intruder

Figure 1 Hybride Encription model [14]

If the attributes are known to the intrusive party, there is a high probability of producing duplicate keys. Similar to how all the data blocks are encrypted when the AES method is used individually. It could cause problems if the hacker is aware of the encryption process. However, the user can choose different-sized keys thanks to AES's various key lengths. In order to prevent intrusions and secure the data, the CP-ABE encryption procedure uses a secure key acquired via AES rather than a random private key. The overall process flow of the suggested hybrid encryption paradigm is shown in Figure1. The selection of attributes for the input file is the first step in the procedure. The selection of attributes for the input file is the first step in the procedure. The attributes are chosen in accordance with user desire and logical pairings. Following the selection of the attributes, encryption is carried out and a set of policies are created for the attributes. A key produced by the AES method is then used to safeguard the file after it has been encrypted for two-tier security. During the decryption procedure, the encrypted file is used, and the AES key is used for authentication. If they do, the operation continues to the next stage; if not, it halts and marks the decryption attempt as an intrusion. The file is decrypted if the key matches the real key; otherwise, attributes are applied; once they match the attributes from the encryption, the file is decrypted. If not, it is designated as an incursion at this level as well. The final file will be plaintext so it may be used in the specified application once it has been encrypted. While writing files to the HDFS system, the encryption process is carried out. Effective file encryption increases data security. Figure2 show how the HDFS system handles encryption. Following is a summary of the steps involved in the encryption process:
Step 1: In the first step, the HDFS client uses the distributed file system to communicate with the master node.
Step 2 A request to create a new file is forwarded to the master node via the distributed file system in step two.
Step 3: The master node selects the accessible data node after determining the data node's space availability.
Step 4. Data node information is shared with the distributed file system and then sent to the HDFS client.
Step 5: The attributes are transferred by the HDFS client before the file is encrypted. The file is encrypted in the data node after the attributes have been chosen.
Step 6: The AES technique is used to generate a key that is then applied to the encrypted file in order to secure it.
Step 7: Using output data streams from a distributed file system, writing starts from a client to a specific data node.
Step 8: Data in the current data node is transferred to another data node if the writing operation has been successful.
Step 9: The master node stores information about the current data node and the replication data node during the replication operation.
Step 10: The distributed file system sends a confirmation to the HDFS client once the data replication in the secondary data node has been successful.
Step 11: When the acknowledgment is received, the HDFS client terminates the writing operation.
Step 12: After getting an acknowledgment from the HDFS client, the writing operation is finally stopped.

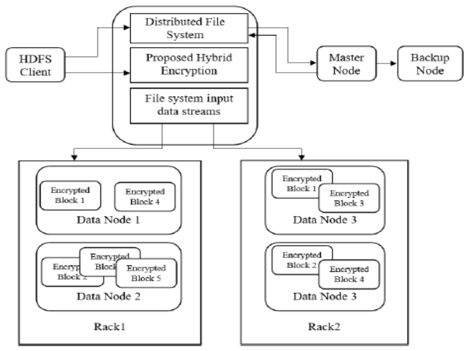


Figure 2 Encryption process in HDFS[14].

The decryption process is carried out as the HDFS system's files are being read. Before reading the operation, the file must be decrypted, and this step aids in detecting any unauthorised access or intruders. Figure.3 shows the HDFS system's decryption procedure. The actions taken throughout the decryption procedure are outlined as follows:
Step 1: The distributed file system is used by the HDFS client to communicate with the master node throughout the decryption process.
Step 2: A request to read a file is forwarded to the master node by a distributed file system.
Step 3: the master node sends information about the data node, which is where the encrypted files are stored.

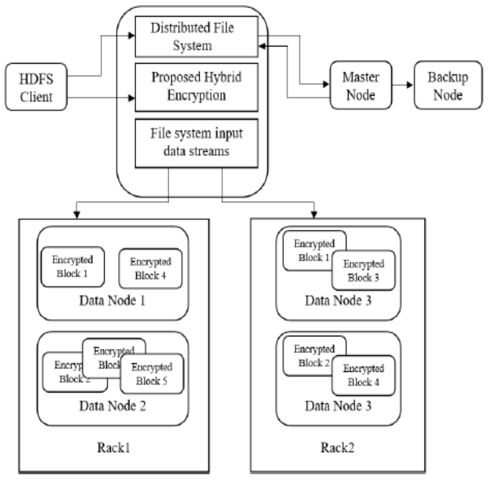


Figure3 Decryption process in HDFS[14].

Step 4: The process begins by the HDFS client using the file system's data input stream, and data from the chosen block is chosen.
Step 5: When a password is entered for authentication, it is compared, and if it matches, the client enters the attributes to decrypt the file.
Step 6: If the matching procedure is failed, the access is marked as a security breach or an unauthorized access.
Sep 7: The HDFS client stops the reading process once the acknowledgment is received.
Step 8: When the HDFS client acknowledges, the reading procedure is finally finished.
When the HDFS client acknowledges, the reading procedure is finally finished. Algorithm 1.
Algorithm 1: A hybrid encryption algorithm for big data security in HDFS.

```
Input: Plain text
Output: Encrypted file
Begin Encryption
Step M: Initialize attributes
Define a set of rules based on the attributes and logical combinations
Apply attributes to the file and perform encryption using a set of policies
Secure file using the key generated by AES
Halt
Begin decryption
Initialize authentication using key matching
If (user input = AES Key)
Allow user to process step N
Else
```

```
Flag as intruder
Step N: perform attribute matching
If (attributes = M)
Decrypt the file
Else
Flag as intruder
Halt
```

IV. RESULT AND DISCUSSION

Experiments carried out in the Hadoop cluster running on an i3 processor running at 2.20 GHz with 8 GB RAM serve to validate the suggested hybrid encryption strategy for large data security in the HDFS environment. Other nodes are chosen as data nodes, and one of the nodes is chosen to serve as the master node. Performance tests for encryption and decryption are conducted using files of various sizes. Comparisons are made between the traditional DES, 3DES, and Blowfish algorithms in terms of throughput, encryption time, decryption time, and efficiency. The simulation parameters used in the proposed work are depicted in Table 1.

Table1. Simulation parameter

Simulation No.	Parameter	Range/Value
1	Input file size	128 MB to 1GB
2	Number of Runs	10
3	Memory	8 GB
4	Key length	128,256 bit
5	Bandwidth	300 MBps

The determination of how long it takes to convert plain text into ciphertext provides the encryption time. Fig. 4 shows that for all files, the suggested model takes the shortest amount of time. The encryption time grows gradually as the size of the file increases, and it takes about 5.5 minutes to encrypt a file with a maximum size of 1 GB of data, compared to 13 minutes for DES, 11.8 minutes for Blowfish, and 12.5 minutes for 3DES. The suggested hybrid encryption method's average encryption time is 7.1 minutes, which is 14 minutes faster than the Blowfish algorithm, 16 minutes faster than the 3DES algorithm, and 8.5 minutes faster than the DES algorithm.

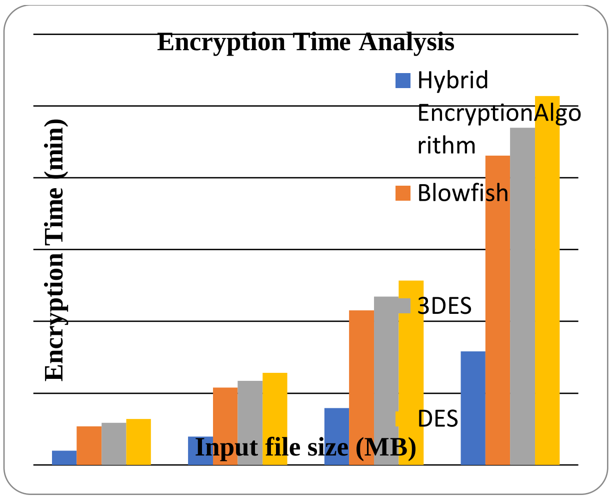


Figure 4 Encryption time analysis.

The time required for converting ciphertext into plain text is used to calculate the decryption time (Figure 5). The investigation reveals that the suggested model Decrypting 1 GB of data takes about 5.1 minutes, which is almost identical to the encryption duration. In contrast to the suggested paradigm, the DES, 3DES, and Blowfish algorithms require 16, 15, and 12.6 minutes, respectively. The suggested solution achieves an average decryption time of 6.5

minutes, which is 23 minutes faster than DES, 20 minutes faster than 3DES, and 15.5 minutes faster than the Blowfish algorithm.

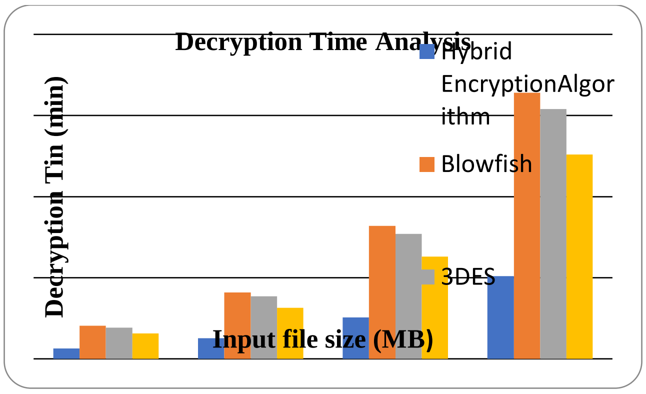


Figure.5 Decryption Time analysis

The throughput is calculated for both the encryption and decryption processes (Figure 6 and 7). For encryption, the throughput is obtained based on the ratio of the size of the text to the time taken to complete the process. The throughputfor decryption is obtained from the ratio of total ciphertext to time taken for the decryption process. It is observed in the analysis that the proposed encryption algorithm obtained maximum throughput compared to conventional algorithms due to minimum text size and computation time. Whereas in conventional encryption algorithms the text size increased and it took more time for the encryption and decryption. Due to this, the throughput for conventional encryption methods was reduced and its performance decreased compared to the Hybrid encryption algorithm. The maximum throughput attained by the proposed model in the encryption and decryption process is 250.25 MB/min and 220.2 MB/min. whereas the throughputs attained by DES, 3DES, and Blowfish algorithms in the encryption process are 75 MB/min, 90 MB/min, 106.4 MB/min, respectively. Similarly, for the decryption process, 70.6 MB/min, 85.1 MB/min, 95.45 MB/min is attained by DES, 3DES, and Blowfish algorithms.

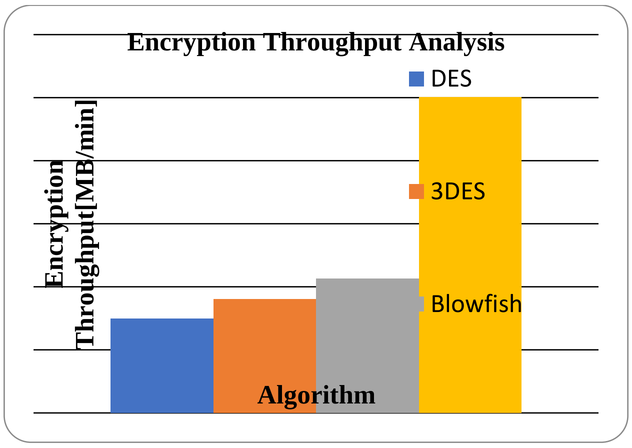


Figure 6 Decryption Throughput

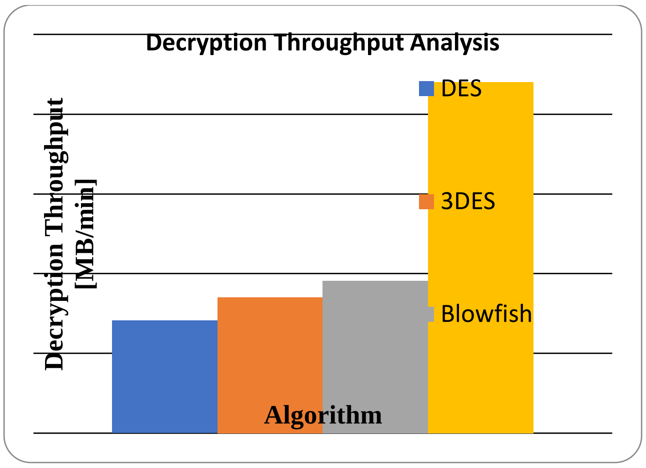
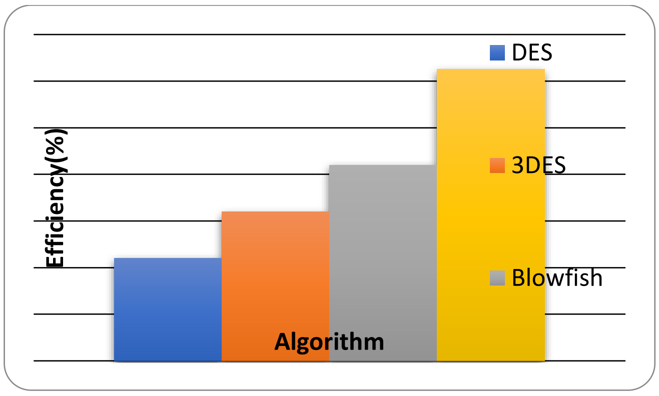


Figure 7 Decryption Throughput

The overall efficiency of the presented system and existing systems depicted in Figure 8 is calculated based on the performance metrics such as encryption time and throughput. The time taken by the system to encrypt 1 GB of data is considered for all the algorithms. Based on the time consumption, intrusion detection, and throughput, the efficiency of the proposed algorithm is calculated.
Mathematically the relationship is formulated into

$$Ef_t = \{t_E, I_{n.}\varphi E\}/\mathrm{d}$$



Figur 8 Efficiency comparison

The results show that the proposed hybrid encryption model attains maximum efficiency compared to other encryption techniques. The increased throughput and minimum computation time for the encryption and decryption process improve the efficiency of the proposed model. At the same time, the efficiency of DES, 3DES, and Blowfish algorithms reduces due to high computation time and low throughput values. The proposed hybrid encryption attains maximumefficiency of 96.5%, whereas DES, 3DES, and Blowfish algorithms attain maximum efficiency of 88.4%, 90.4%, 92.4%, respectively which is much smaller than the proposed model efficiency.Table 2 depicts the overall performance comparative analysis of the proposed model and conventional DES, 3DES, and Blowfish algorithms.

Table 2 Performance Comparative Analysis

Parameters	DES	3DES	Blowfish	Hybrid Encryption Algorithm
Encryption Time	25.69	23.48	21.54	7.92
Decryption Time	3018	27.11	22.14	6.51
Encryption Throughput	75.02	90.04	106.42	250.25
Decryption Throughput	70.64	85.16	95.45	220.22
Efficiency %	88.4	90.4	92.4	96.5

The average values for all the parameters are listed in Table 1. The results show that the presented hybrid encryption algorithm performs better than conventional algorithms in all aspects from small-size files to large-size files. Thus, it improves the big data security in the HDFS environment.

V. CONCLUSION

In this study, a hybrid encryption algorithm for the Hadoop distributed file system environment's big data security was introduced. Hybrid encryption, which combines the CP-ABE and AES algorithms, offers two-tier encryption for files being generated in the HDFS data node. The input file's specified characteristics were encrypted utilising the CP-ABE. Furthermore, the password for the encrypted file was created using the key generated by the sophisticated encryption standard method, which enhances data security and identifies the hacker throughout the decryption process. Throughput, efficiency, and encryption and decryption times were used to validate the suggested model's performance. In comparison to the traditional DES, 3DES, and Blowfish algorithms, the presented hybrid encryption model achieved higher performance.

REFERENCES

[1] P.K. Mallepalli, S.R. Tumma, A lightweight hybrid scheme for security of big data, Materials Today: Proceedings, pp. 1–14, 2021, doi: 10.1016/j.matpr.2021.03.151.
[2] W. Gao, W. Yu, F. Liang, W.G. Hatcher, C. Lu, Privacy-preserving auction for big data trading using homomorphic encryption, IEEE Transactions on Network Science and Engineering, 7(2): 776–791, 2020, doi: 10.1109/TNSE.2018.2846736.
[3] A. Alabdulatif, I. Khalil, X. Yi, Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption, Journal of Parallel and Distributed Computing, 137: 192–204, 2020, doi: 10.1016/j.jpdc.2019.10.008.
[4] K. Sharma, A. Agrawal, D. Pandey, R.A. Khan, S.K. Dinkar, RSA based encryption approach for preserving confidentiality of big data, Journal of King Saud University – Computer and Information Sciences, pp. 1–16, 2019, doi: 10.1016/j.jksuci.2019.10.006.
[5] S. Tahir, L. Steponkus, S. Ruj, M. Rajarajan, A. Sajjad, A parallelized disjunctive query based searchable encryption scheme for big data, Future Generation Computer Systems, 109: 583–592, 2020, doi: 10.1016/j.future.2018.05.048.
[6] D. Puthal, X. Wu, N. Surya, R. Ranjan, J. Chen, SEEN: A selective encryption method to ensure confidentiality for big sensing data streams, IEEE Transactions on Big Data, 5(3): 379–392, 2019, doi: 10.1109/TBDATA.2017.2702172.
[7] P. Perazzo, F. Righetti, M. La Manna, C. Vallati, Performance evaluation of attributebased encryption on constrained IoT devices, Computer Communications, 170: 151–163, 2021, doi: 10.1016/j.comcom.2021.02.012.
[8] H. Deng, Z. Qin, Q. Wu, Z. Guan, Y. Zhou, Flexible attribute-based proxy re-encryption for efficient data sharing, Information Sciences, 511: 94–113, 2020, doi: 10.1016/j.ins. 2019.09.052.
[9] P.S. Challagidad, M.N. Birje, Efficient multi-authority access control using attributebased encryption in cloud storage, Procedia Computer Science, 167: 840–849, 2020, doi:10.1016/j.procs.2020.03.423.

[10] S. Aditham, N. Ranganathan, A system architecture for the detection of insider attacks in big data systems, IEEE Transactions on Dependable and Secure Computing, 15(6): 974–987, 2018, doi: 10.1109/TDSC.2017.2768533.
[11] J.S. Raj, A novel encryption and decryption of data using mobile cloud computing platform, IRO Journal on Sustainable Wireless Systems, 2(3): 118–122, 2021, doi: 10.36548/ jsws.2020.3.002.
[12] S. Shakya, S. Smys, Big data analytics for improved risk management and customer segregation in banking applications, Journal of ISMAC, 3(3): 235–249, 2021, doi: 10.36548/ jismac.2021.3.005.
[13] T. Mohanraj, R. Santhosh, "Hybrid Encryption Algorithm for Big Data Security in the Hadoop Distributed File System", CAMES, 29(1–2), 33–48, 2022, doi: 10.24423/cames.375