CAMES, **29**(1–2): 33–48, 2022, doi: 10.24423/cames.375

This article belongs to the *Special Issue on Advanced Optimization Methods for Uncertainties in Intelligent Industrial Systems* edited by Dr. I.J. Jacob, Dr. B.H. bin Ahmad and Dr. Z.F. Khan

Hybrid Encryption Algorithm for Big Data Security in the Hadoop Distributed File System

T. MOHANRAJ*, R. SANTHOSH

Department of Computer Science and Engineering
Faculty of Engineering, Karpagam Academy of Higher Education
Coimbatore, Tamil Nadu, India
e-mail: santhoshrd@gmail.com
*Corresponding Author e-mail: mohanrajt.me@gmail.com

A large amount of structured and unstructured data is collectively termed big data. The recent technological development streamlined several companies to handle massive data and interpret future trends and requirements. The Hadoop distributed file system (HDFS) is an application introduced for efficient big data processing. However, HDFS does not have built-in data encryption methodologies, which leads to serious security threats. Encryption algorithms are introduced to enhance data security; however, conventional algorithms lag in performance while handling larger files. This research aims to secure big data using a novel hybrid encryption algorithm combining cipher-text policy attribute-based encryption (CP-ABE) and advanced encryption standard (AES) algorithms. The performance of the proposed model is compared with traditional encryption algorithms such as DES, 3DES, and Blowfish to validate superior performance in terms of throughput, encryption time, decryption time, and efficiency. Maximum efficiency of 96.5% with 7.12 min encryption time and 6.51 min decryption time of the proposed model outperforms conventional encryption algorithms.

Keywords: big data security, Hadoop, data encryption and decryption, Hadoop distributed file system (HDFS).

1. Introduction

Rapid technology development generates large amount of data every second. Numerous applications, sensor networks, social networking sites, and small to large-scale organizations process this vast data [1, 2]. Any complex, diverse and large volume of data that conventional data processing applications cannot process is collectively termed big data. Originally, volume, variety, and velocity were the properties considered to define big data. Later, this has been extended with a few other properties such as veracity, venue, validity, vocabulary, vagueness, and value to make counterpart descriptions of big data. However, these features usually encounter parallel issues while storing, analyzing, and extracting the desired results. Big data analytics is the process of analyzing a large amount of complex data to identify correlations and expose hidden patterns. However, there is a contradiction between big data security and data widespread use [3].

Hadoop is an open-source framework that can process and store big data sets in a distributed, reliable, and scalable computing platform [4]. The cost-efficient, quick processing, fault-tolerant, flexible Hadoop is commonly used in public cloud services or large clusters. Hadoop's two main components are the Hadoop distributed file system (HDFS) and MapReduce [5]. Large volumes of structured or unstructured data can be processed in parallel on MapReduce. Multiple hard drives logical file systems are used in HDFS to distribute large data. Though the processing ability of Hadoop is much better than conventional data processing systems, it has flaws in its security features. The Hadoop design is not meant for security, so it does not provide any security scheme to protect data in storage or transit. Big data is used by many organizations for research and marketing, but they may not concentrate on security perspectives [6]. The data breach would result in serious reputational damage and legal repercussions.

The Hadoop big data handles sensitive information such as financial data, personal information, corporate data, and confidential information such as customer, client, and employee details. Moreover, organizations store and analyze a large amount of data that must be secured in HDFS storage through encryption, threat detection, and logging procedures. These procedures support the systems in detecting the threats in the early stages and secure the user data. The major characteristic of big data is diversity. The data type of big data may be text, image, audio, video, etc. Recently, various techniques have evolved to protect the data from generation phase to storage phase. Data privacy is enhanced in the generation phase by using data falsification techniques and restricting access. In the storage phase, data security and privacy are provided mainly based on encryption techniques [7].

Encryption is a process of transforming plain text into ciphertext. The conversion of explicable data into inexplicable form ensures data privacy by allowing the users to have proper authentication [8] and blocking others. Encryption keys are exchanged, and the same will be used in the decryption process. For strong data protection and confidentiality, encryption is still in use. Encryption algorithms are classified into the symmetric key algorithms and public key algorithms. Existing research works use numerous encryption algorithm such as AES, RSA, DES, and ABE to secure big data [9, 10]. However, the performance of this convention encryption technique declines while handling a dynamic large amount of data. So it is essential to develop an efficient data encryption algorithm for big data security.

The major contributions of this research work are summarized as follows:

- A hybrid data encryption algorithm is presented using CP-ABE and AES.
- Comparative evaluation of the proposed encryption algorithm with traditional encryption algorithms such as DES, 3DES, and Blowfish is performed.

The rest of the article is arranged as follows: a brief literature analysis is presented in Sec. 2; the proposed hybrid encryption algorithm is presented in Sec. 3; experimental results are presented in Sec. 4, and finally, conclusion in Sec. 5.

2. Related works

Brief literature analyses of big data security and encryption techniques are presented in this section. Researchers pay significant attention to improve big data applications' security and privacy features using various encryption techniques [11–13]. The homomorphic cryptography and secure network protocol design reported in [14] uses a privacy-preserving auction scheme to improve the data confidentiality and trust between the user and third-party service provider. The homomorphic encryption scheme is used to secure the data communication between the user and third-party service providers. The enhanced strategy includes signature-based verification for improved data security. However, the performance of the system lags if the number of users and file size are increased.

The fully homomorphic encryption model reported in [15] overcomes the internal and external data privacy breaches and threats in the big data environment. The powerful cryptosystem analyzes the tasks and partitions the computation and data into two different subsets. This process rapidly accelerates the data processing ability of the system and attains a high level of accuracy. In [16], an acceleration procedure for complex data encryption and computation is reported to handle a large number of scenarios. For intensive data encryption, an adaptive crypto acceleration secure data storage mechanism that dynamically improves the big data file operation modes is presented. The research work attains a better tradeoff compared to other software and hardware accelerators.

The data analysis model reported in [17] efficiently handles the privacy issues in the health information exchange process through an encryption algorithm. The privacy issues in health information and the necessity of data encryption are addressed through patient-centric data access control mode. The proposed RSA-based encryption encrypts the patient file and transmits using several domains. Though the encryption model effectively handles the security and privacy issues, it requires multiple domains to transmit data. This increases the system cost compared to other data transmission methods.

The findings in [18] report the difficulties faced by the user when the data is outsourced. Data privacy and confidentiality are the primary objectives of the research work, and they are improved using multi-keyword searchable encryption. The probabilistic trapdoors are formed to resist attacks and improve data security. The selective encryption method reported in [19] ensures data confidentiality in big data streams. The trustworthiness of collected data depends on the security features such as data integrity and confidentiality. The authors use the selective encryption method to increase the data stream encryption and decryption performance without compromising data integrity and confidentiality.

Attribute-based encryption reported in [20] discussed the challenges in conventional cryptography techniques. The encryption model addresses the importance of fine-grained access control in a big data environment. The flexible policies enhance the access control on encrypted data and provide more feasibility while handling a large amount of data. The performances are evaluated based on ten different attributes and adopted cryptographic acceleration to attain better performance. Minimum memory and power are the major merits of this encryption model. The hybrid attribute-based encryption (ABE) model reported in [21] overcomes the limitation of conventional ABE algorithms. Since the policies defined by the conventional models become obsolete over a particular time, the reported hybrid model includes proxy re-encryption to convert ABE's ciphertext into identity-based encryption (IDE) ciphertext. The compact utilization of identity-based encryption and key randomization improves security and avoids collisions in the data.

The authors in [22] investigated the unauthorized data access issues and privacy challenges in big data cloud using the CP-ABE algorithm. The necessity of multi-authority access control and data privacy were addressed by the encryption algorithm. User data and fine-grained access were provided by the role hierarchy algorithm and hierarchy access structure. Computation time and minimum storage consumption are the major merits of the hierarchy algorithm.

Attack detection and intrusion detection are important factors to be considered in big data security analysis as they affect the data confidentiality and privacy. Various attack detection models and intrusion detection techniques have evolved to improve data security. A two-step attack detection algorithm reported in [23] secures communication protocol through instruction sequences. These instruction sequences are further matched with the nodes to analyze the system needs. The encryption and decryption procedure reported in [24] considers the issues in attribute-based encryption pairing operations. which reduce the encryption system performance. To overcome it, a lightweight fine-grain data sharing methodology is adopted for outsourced data operations, enhancing the overall data security and avoiding decryption key exposure.

The analysis shows that cryptographic techniques have key management and authentication procedures that are quite difficult for complex data management. Though homomorphic encryption provides better data security, the encryption process is incredibly slow and non-performant. Whereas fully homomorphic encryption provides better data usability and privacy. However, the slow computation speed or accuracy issues make the technique infeasible for big data applications. RSA-based encryption provides better authenticity and confidentiality, however it is slow while handling big data. Conventional encryption algorithms perform well for normal data. However, for big data, due to the large amount and data diversity, the computation time increases for the conventional encryption algorithms. If hybrid encryption models are considered, the encryption performance is improved, but it can be further improved to attain a better system. Based on this finding, it is observed that hybrid encryption techniques can be a better choice to improve big data security. Considering these observations, our research proposes a hybrid encryption model to obtain maximum performance with minimum computation cost for big data security in a distributed file system environment.

3. Proposed work

The proposed hybrid encryption algorithm comprises CP-ABE and AES algorithm to secure the HDFS big data. The double encryption procedure provides better data security compared to traditional encryption techniques. Recently, ABE has gained more attention due to its decentralized access control and secure communication abilities in dynamic conditions. However, a user cannot define any policies or procedures to define the encryption process. To provide more user access control over the encryption process, access control policies are introduced in terms of ciphertext policies.

Along with attributes, users can define the policy for the encryption and decryption process. Also, these access control policies cryptographically secure the data in transmission and storage. User requirements are efficiently handled in ABE and simplify the private and public keys concept into a single attribute-based concept. The attribute policies defined in ABE are obtained based on the attribute logical combinations. Based on the user needs and application, the attributes and their logical combinations can be framed in ABE. The proposed hybrid encryption scheme can handle static and dynamic attributes (Fig. 1).

In CP-ABE, the encryption is performed only for the attributes and not on the entire block. AES is used to produce a 128-bit key, and it is used by secret or symmetric ciphers in the encryption and decryption process. The two-step process initially executes the ABE algorithm and then AES is employed. Instead of a hybrid approach, if the algorithms are executed individually, CP-ABE

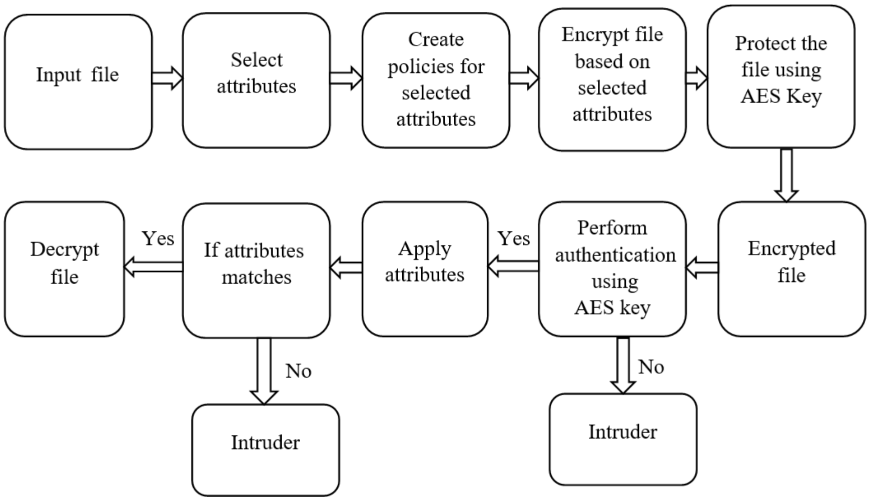
Input file
Select attributes
Create policies for selected attributes
Encrypt file based on selected attributes
Protect the file using AES Key
Encrypted file
Perform authentication using AES key
Yes
Apply attributes
If attributes matches
Yes
Decrypt file
No
Intruder
No
Intruder

Fig. 1. Overview of the proposed hybrid encryption model.

relies on some random user private key, which is needed to be obtained based on attributes. If the attributes are known to the intruder then the chances for generating duplicate keys are high.

Similarly, if the AES algorithm is applied individually, all the data blocks are encrypted in a similar manner. which might lead to issues if the intruder knows the encryption procedure. However, the different key lengths of AES allow the user to select different-sized keys. So instead of using a random private key, a secure key obtained by AES is used in the CP-ABE encryption process to avoid intrusions and secure the data. Figure 1 depicts the overall process flow of the proposed hybrid encryption model.

The process starts from the selection of attributes for the input file. The attributes are selected based on the user preference and logical combinations. Upon selecting the attributes, a set of policies are framed for the selected attributes, and encryption is performed. Once the file is encrypted for two-tier security, a key generated by the AES algorithm is used to secure the file. The encrypted file is used in the decryption process, where the authentication is performed using the AES key. If it matches, then it passes into the next stage; otherwise, the decryption process stops and flags the decryption attempt into intrusion. If the key matches with the actual key, then attributes are applied, and once it matches with the attributes from the encryption, then the file is decrypted. Otherwise, in this stage also it is marked as an intrusion. The final decrypted file will be plaintext that can be used in the requested application.

The encryption process is performed while writing the files in the HDFS system. Data security is enhanced by effectively encrypting the files. The encryption process in the HDFS system is depicted in Fig. 2. The steps performed in the encryption process are summarized as follows:

Step 1: In the first step, the HDFS client communicates with the master node using the distributed file system.

Step 2: Distributed file system forwards the request to the master node that includes a request to create a new file.

Step 3: The master node checks the space availability of the data node and selects the available data node.

Step 4: Details of data nodes are shared with the distributed file system and then communicated to the HDFS client.

Step 5: The HDFS client transfers the attributes before encrypting the file. Once the attributes are selected, the file is encrypted in the data node.

Step 6: The encrypted file needs to be secured; for that, a key is generated using the AES algorithm and applied to the file.

Step 7: The writing process starts from client to specific data node using distributed file system output data streams.

Step 8: If the writing process is completed, then the data in the current data node is transferred to another data node.

Step 9: During the replication process, the master node holds the details of current data node and replication data node.

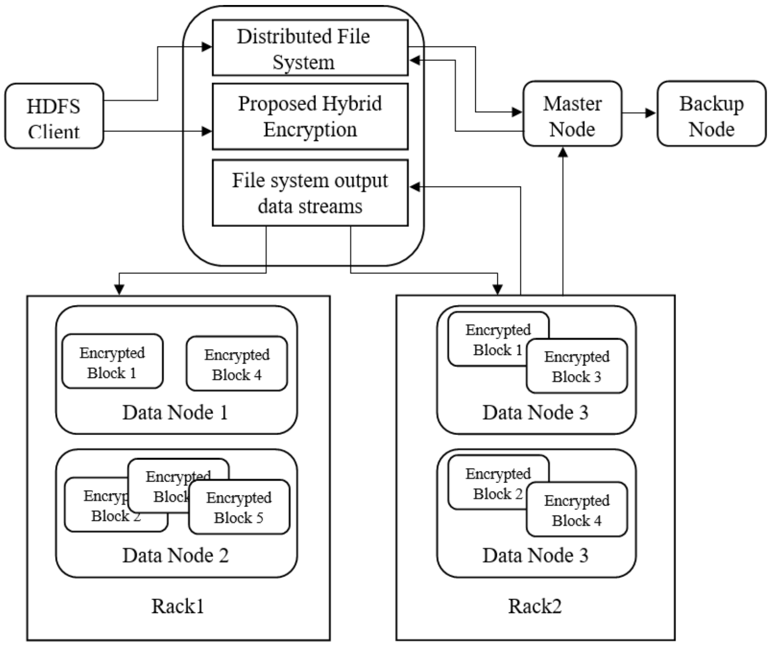

FIG. 2. Encryption process in HDFS.

Step 10: Once the data is successfully replicated in the secondary data node, an acknowledgment is sent to the HDFS client through the distributed file system.

Step 11: The HDFS client stops the writing process once the acknowledgment is received.

Step 12: Finally, writing operation is terminated upon receiving the HDFS client acknowledgment.

The decryption process is performed while reading the files in the HDFS system. It is essential to decrypt the file before reading the operation and this process helps to identify the unauthorized access or intruder. The decryption process in the HDFS system is depicted in Fig. 3. The steps performed in the decryption process are summarized as follows:

Step 1: In the decryption process, the HDFS client communicates the master node using the distributed file system.

Step 2: Distributed file system forwards the request to the master node that includes a request to read a file.

Step 3: The master node transfers the details of the data node, where the encrypted files are stored.

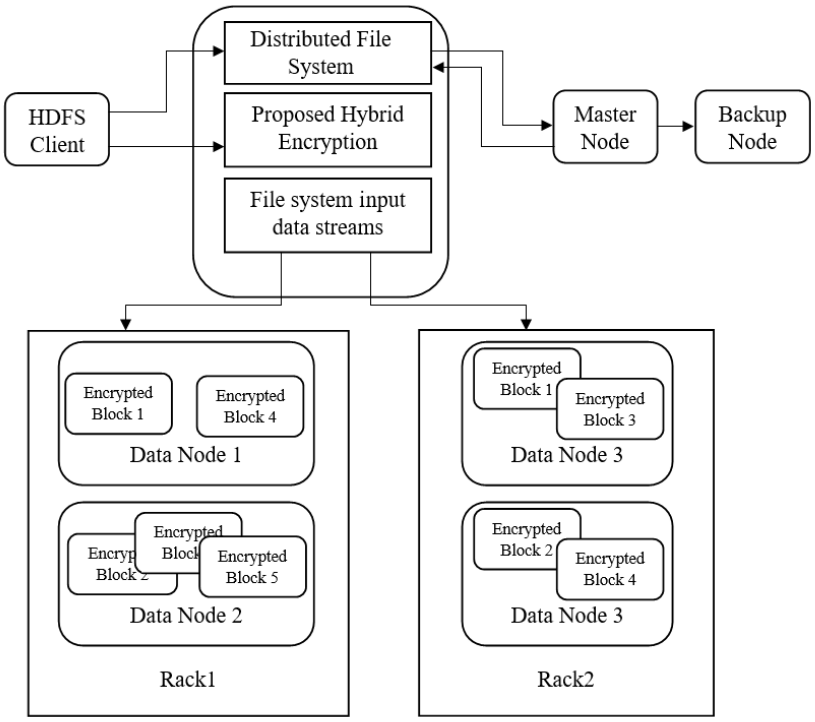

Fig. 3. Decryption process in HDFS.

Step 4: The HDFS client initiates the process through the file system data input stream, and data from the specified block is selected.

Step 5: For authentication, a password match is performed, and if it matches, the client will enter the attributes to decrypt the file.

Step 6: If the matching process fails, then access is flagged as an intrusion or unauthorized access.

Sep 7: The HDFS client stops the reading process once the acknowledgment is received.

Step 8: Finally reading operation is terminated upon receiving the HDFS client acknowledgment.

As shown in the above steps, the encryption and decryption process of the proposed model provides enhanced data security to big data in the HDFS environment of writing and reading operations. The summarized pseudocode for the proposed data security model is given in 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
```

4. Results and discussion

The proposed hybrid encryption model for big data security in the HDFS environment is verified through experiments performed in the Hadoop cluster installed in the i3 processor 2.20 GHz with 8 GB RAM. One of the nodes is selected as the master node and other nodes are selected as data nodes. Files with different sizes are used to evaluate the encryption and decryption performance. Parameters such as throughput, encryption time, decryption time, and efficiency are compared with conventional DES, 3DES, and Blowfish algorithms. The simulation parameters used in the proposed work are depicted in Table 1.

Table 1. Simulation parameters.

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

The encryption time is obtained by calculating the time taken to generate a ciphertext from plain text. It is observed in Fig. 4 that the time taken for the proposed model is minimum for all the files. The encryption time increases gradually from small to large file size, and for a maximum file size of 1 GB of data, the encryption time is approximately 5.5 min, whereas DES takes 13 min, Blowfish takes 11.8 min and 3DES takes 12.5 min to encrypt the same size of the file. The average encryption time of the proposed hybrid encryption algorithm is 7.1 min, which is 14 min less than the Blowfish algorithm, 16 min less than the 3DES algorithm, and 8.5 min less than the DES algorithm.

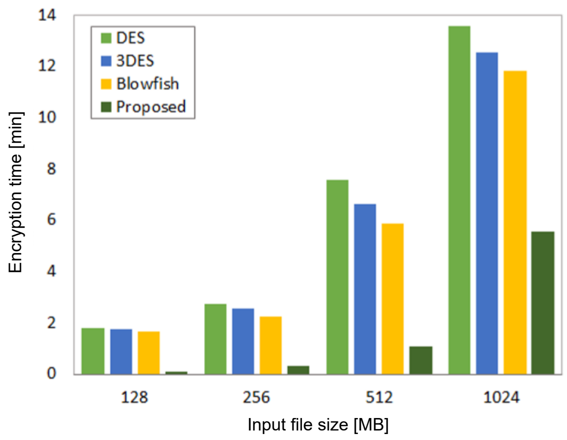

Fig. 4. Encryption time analysis.

The decryption time is calculated for the time taken to convert ciphertext into plain text (Fig. 5). In the analysis, it is observed that the proposed model decryption time is significantly smaller compared to other encryption algorithms. Almost similar to encryption time, the decryption time also consumes approximately 5.1 min to decrypt 1 GB of data. Whereas DES, 3DES, and Blowfish algorithms take 16.4, 15.4, and 12.6 min, respectively, which is much higher than the proposed model. The average decryption time attained by the proposed approach is 6.5 min, which is 23 min less than DES, 20 min less than 3DES, and 15.5 min less than the Blowfish algorithm.

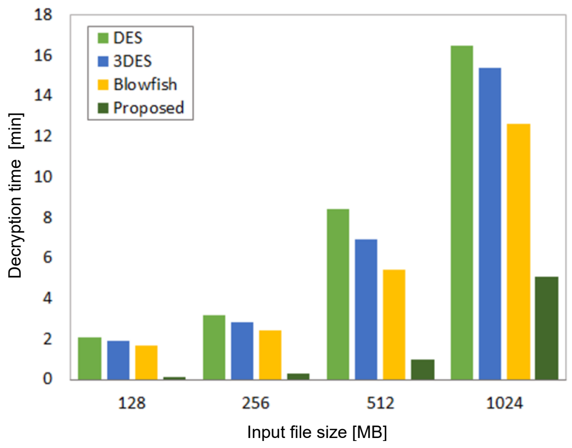

Fig. 5. Decryption time analysis.

The throughput is calculated for both the encryption and decryption processes (Figs 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 throughput for 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 en-

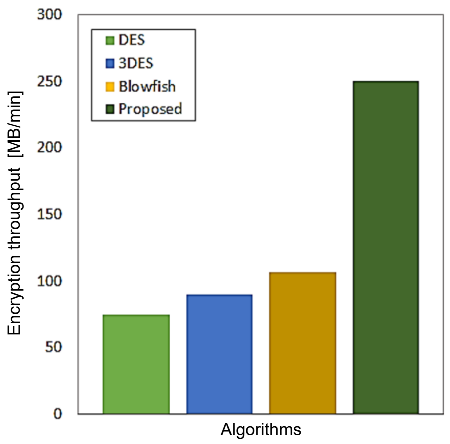

Fig. 6. Encryption throughput analysis.

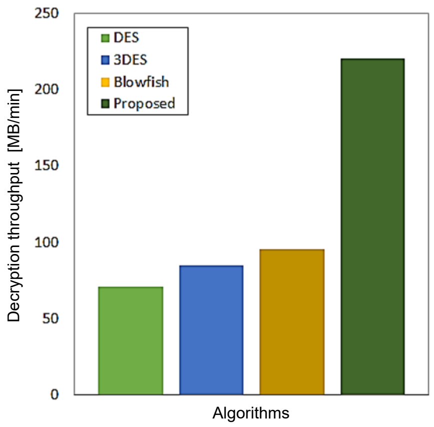

Fig. 7. Decryption throughput analysis.

cryption 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 proposed 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.

The intrusion detection efficiency of the proposed model and existing models is verified for different instances, and the performances are comparatively depicted in Fig. 8. All ten instances are obtained by modifying the encrypted data

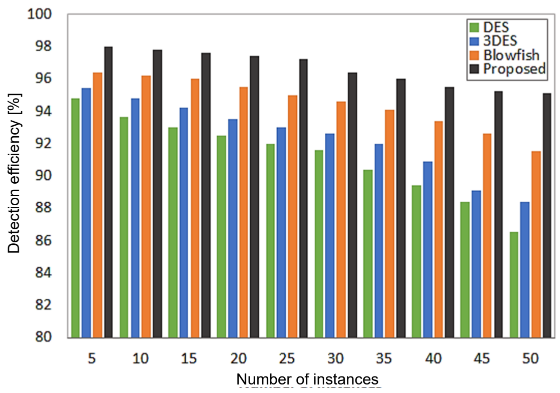

Fig. 8. Intrusion detection efficiency.

in the system, and modifications are performed by varying the variables in the encrypted text. The modifications are performed in the initial portion of encrypted files for a few instances and modified in the middle and last portions of encrypted files for a few instances. So that the response of proposed and existing models can be observed for different intrusions. Initially, the results show that the detection efficiency is maximum and by decrypting the text and matching with the actual text the deviations are identified as intrusions. However, changes made in the middle and last portions take more time to detect the modifications as the decryption is performed block by block. Due to this, the detection time is initially high, which leads to a decreased detection efficiency. Compared to all the other models, the detection efficiency of the proposed model attains better results due to the hybrid encryption and decryption procedures.

The overall efficiency of the proposed system and existing systems depicted in Fig. 9 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

$$Ef_t = \{t_E, I_\eta, \varphi_E\} \,|d, \tag{1}$$

where t_E represents the encryption time, I_η represents the intrusion detection efficiency, φ_E represents the encryption throughput, and d represents the data.

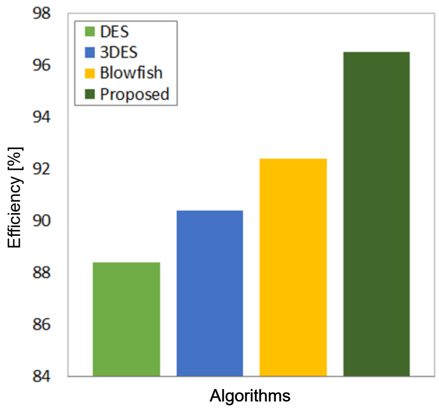

Fig. 9. Efficiency comparison.

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 maximum efficiency 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	Proposed
Encryption time [min]	25.69	23.48	21.54	7.12
Decryption time [min]	30.18	27.11	22.14	6.51
Encryption throughput [MB/min]	75.02	90.04	106.42	250.25
Decryption throughput [MB/min]	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 proposed 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.

5. Conclusion

A hybrid encryption algorithm for big data security in the Hadoop distributed file system environment was presented in this research. CP-ABE and AES algorithm were combined as hybrid encryption to provide two-tier encryption for the files while writing in the HDFS data node. The CP-ABE was used to encrypt the attributes selected for the input file. Further, the key generated by the advanced encryption standard algorithm was used as a password for the encrypted file, which enhances the data security and identifies the intruder in the decryption process. The performance of the proposed model was validated in terms of encryption time, decryption time, throughput, and efficiency. The proposed hybrid encryption model attained better performance compared to conventional DES, 3DES, and Blowfish algorithms. However, the CP-ABE scheme relies on policies and attributes and if the attributes are not appropriately selected, then there might be deviations in the performances on different runs, which is the minor limitation observed in the proposed work. Further, this research work can be extended by introducing optimization techniques to improve other performance metrics.

References

1. 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, doi: 10.26599/BDMA.2020.9020006.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. M. Parihar, Big Data security and privacy, *International Journal of Engineering Research & Technology*, **10**(07): 323–327, 2021.

13. R. Chatterjee, R. Chakraborty, J.K. Mondal, Design of lightweight cryptographic model for end-to-end encryption in IoT domain, *IRO Journal on Sustainable Wireless Systems*, **1**(4): 215–224, 2019, doi: 10.36548/jsws.2019.4.002.

14. 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.

15. 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.

16. C. Xiao, P. Li, L. Zhang, W. Liu, N. Bergmann, ACA-SDS: Adaptive crypto acceleration for secure data storage in big data, *IEEE Access*, **6**: 44494–44505, 2018, doi: 10.1109/ACCESS.2018.2862425.

17. 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.

18. 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.

19. 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.

20. P. Perazzo, F. Righetti, M. La Manna, C. Vallati, Performance evaluation of attribute-based encryption on constrained IoT devices, *Computer Communications*, **170**: 151–163, 2021, doi: 10.1016/j.comcom.2021.02.012.

21. 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.

22. P.S. Challagidad, M.N. Birje, Efficient multi-authority access control using attribute-based encryption in cloud storage, *Procedia Computer Science*, **167**: 840–849, 2020, doi: 10.1016/j.procs.2020.03.423.

23. 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.

24. 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.

25. 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.

Received August 8, 2021; revised version November 2, 2021; accepted November 10, 2021.