

Enhance big data security based on HDFS using the hybrid approach

Fayçal Zine-Dine¹, Sara Alcabnani¹, Ahmed Azouaoui², Jamal El Kafi¹

¹Department of Computer Science, Faculty of Sciences, Chouaib Doukkali University, El Jadida, Morocco

²High School of Technology EST, University Ibn Tofail, Kenitra, Morocco

Article Info

Article history:

Received Mar 28, 2024

Revised Oct 8, 2024

Accepted Oct 30, 2024

Keywords:

Big data

Data security

Decryption

Encryption

Hadoop

HDFS

ABSTRACT

Hadoop has emerged as a prominent open-source framework for the storage, management, and processing of extensive big data through its distributed file system, known as Hadoop distributed file system (HDFS). This widespread adoption can be attributed to its capacity to provide reliable, scalable, and cost-effective solutions for managing large datasets across diverse sectors, including finance, healthcare, and social media. Nevertheless, as the significance and scale of big data applications continue to expand, the challenge of ensuring the security and safeguarding of sensitive data within Hadoop has become increasingly critical. In this study, the authors introduce a novel strategy aimed at bolstering data security within the Hadoop storage framework. This approach specifically employs a hybrid encryption technique that leverages the advantages of both advanced encryption standard (AES) and data encryption standard (DES) algorithms, whereby files are encrypted in HDFS and subsequently decrypted during the map task. To assess the efficacy of this method, the authors performed experiments with various file sizes, benchmarking the outcomes against other established security measures.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Fayçal Zine-Dine

Department of Computer Science, Faculty of Sciences, Chouaib Doukkali University

El Jadida, Morocco

Email: zdfaycal@gmail.com

1. INTRODUCTION

The processing of big data has become essential across various fields, with apache Hadoop emerging as a leading platform for distributed storage and processing [1], [2]. Its Hadoop distributed file system (HDFS) facilitates the effective management of extensive datasets [3], [4]. Nonetheless, ensuring the security of data within HDFS presents a considerable challenge, as the existing security features are insufficient to safeguard sensitive information against unauthorized access or cyber threats [5]. Additionally, big data is characterized by several inherent challenges, commonly referred to as the 5 V's: volume, velocity, variety, value, and veracity [6], [7].

Hadoop [8] is widely acknowledged as a powerful framework for executing applications on large clusters constructed from standard hardware. As an open-source framework for distributed computing, developed in Java, Hadoop comprises two fundamental components. The first is HDFS, a distributed file system that guarantees reliable, scalable, and fault-tolerant data storage across commodity hardware, capable of accommodating diverse data formats, including text, images, and videos. The second component is MapReduce, a programming model designed to efficiently process and analyze large datasets in parallel across multiple nodes within a computing cluster. This model consists of two distinct tasks: the "Map" task,

which processes and converts input data into intermediate key-value pairs, and the "Reduce" task, which consolidates and summarizes these intermediate results [9], [10].

Numerous research initiatives have concentrated on enhancing data security within the HDFS. For example, O'Malley *et al.* [5] proposed a security framework for Hadoop that integrates Kerberos and SSL, thereby improving authentication and access control mechanisms. In another study, Park and Lee [11] presented a method that employs advanced encryption standard (AES) for encrypting HDFS data, which effectively lowers computational overhead. Additionally, Mahmoud in [12] examined a strategy for big data security utilizing HDFS, which combines AES with stream ciphers such as one-time pad (OTP) to bolster both security and encryption efficiency.

Nevertheless, these existing methodologies encounter several challenges. Certain solutions may demand significant computational resources, potentially hindering system performance, while others might not achieve the desired level of security. Furthermore, the larger size of encrypted files can impose additional burdens on storage and bandwidth, adversely affecting the overall efficiency of the Hadoop ecosystem. The pursuit of a high-performance, efficient, and resilient encryption solution for HDFS data continues to be a significant challenge. Current methodologies frequently grapple with the trade-offs between performance, security, and the size of encrypted files. Some encryption algorithms are notably resource-intensive, resulting in slower read and write operations within HDFS. Achieving optimal security necessitates robust defenses against known threats and stringent key management practices [13]. The increased size of encrypted files can exacerbate storage and bandwidth constraints, further diminishing the overall performance of the Hadoop system.

This study introduces a hybrid encryption strategy that integrates the AES and the data encryption standard (DES) for securing data within the HDFS. The primary objective of this approach is to bolster data security while enhancing encryption efficiency and minimizing the size of the encrypted files. The implementation of AES in cipher block chaining (CBC) mode facilitates effective data encryption, whereas DES is utilized to encrypt the AES key, thereby adding an extra layer of security. The advantages of this method include: improved performance due to the rapid encryption capabilities of AES in CBC mode, enhanced security through a dual-layer encryption mechanism, and a reduction in file size achieved by employing DES for key encryption, which alleviates storage and bandwidth demands.

This paper elaborates on our hybrid encryption methodology for HDFS. In section 2, we provide a thorough review of existing literature, analyzing various strategies that have been developed to secure data within HDFS. Section 3 outlines the architecture of our proposed solution, emphasizing the synergy between the AES and DES algorithms. Section 4 details the experimental results obtained to assess the performance of our method, focusing on encryption duration, file size, and throughput, and compares these metrics with those of current methodologies. Finally, section 5 summarizes the findings of our research.

2. RELATED WORK

The Hadoop framework lacks built-in security measures for the diverse types of data it stores, prompting various researchers to propose multiple strategies for securing files within the HDFS. Among these strategies, encryption emerges as a prominent solution for safeguarding data stored in DataNodes and ensuring secure data transmission during MapReduce operations. In one study [5], a secure HDFS architecture is presented that employs Kerberos over SSL to facilitate robust mutual authentication and access control, thereby enhancing the overall security of HDFS. Another research effort [14] introduces a dependable file system architecture that integrates authentication agent technology with fully homomorphic encryption, offering enhanced reliability and security against threats from data, hardware, users, and operational processes.

Zhou and Wen [15] apply cipher text policy and attribute-based encryption (CP_ABE) to establish access control credentials for authorized cloud users, utilizing an encrypted data access control framework rather than relying on individual user identities. Additionally, Park and Lee [11] propose a method for securing Hadoop through encrypted HDFS by incorporating encryption/decryption functions based on the AES algorithm. Their experiments indicate that the computational overhead is reduced by less than 7% during representative MapReduce jobs.

Furthermore, the researchers in [16] present a hybrid encryption approach for HDFS that combines HDFS-RSA and HDFS-pairing. However, it is noted that both the read and write performance of the encrypted HDFS is inferior to that of the standard HDFS. In the research conducted by Yang *et al.* [17], a novel triple encryption scheme has been proposed, which is integrated into Hadoop for cloud data storage. This scheme incorporates hybrid encryption techniques, specifically utilizing RSA and DES algorithms alongside IDEA to secure the user's RSA private key. This method ensures that files stored in HDFS are encrypted while they are temporarily held in a buffer following their upload to HDFS.

The study referenced in [18] discusses various strategies aimed at enhancing information security and addressing safety concerns through methodologies such as security hardening and attribute selection. Notably, the researcher has put forth a security-oriented approach to data collection and threat analytics, which aims to mitigate risks prior to the occurrence of security breaches. In the work of Maheswari [19] presented, a strategy to bolster security within Hadoop is outlined. This approach employs AES in conjunction with message digest (MD5) and DES algorithms for data encryption. Additionally, the digital signature algorithm (DSA) is utilized for data authentication purposes. In this framework, authorized users upload their data to cloud storage, where it is encrypted before being stored in HDFS, allowing for subsequent sharing among multiple recipients.

Conversely, Mahmoud [12] have introduced a methodology utilizing HDFS aimed at decreasing both the computation time and size of encrypted files through the implementation of AES and OTP algorithms. Additionally, they employed the AES algorithm in conjunction with the cipher block chaining mode, specifically the electronic codebook (ECB) mode, which is particularly suitable for managing HDFS blocks, while the OTP algorithm functions as a stream cipher.

3. SECURE HADOOP USING ON AES AND DES

3.1. Overview

The DES is a widely recognized block cipher algorithm, primarily due to its role as a standard for symmetric key encryption [20], [21]. Originally developed by IBM in the 1970s, it was officially adopted by the United States government in 1976 as a Federal Information Processing standard (FIPS). DES operates on 64-bit blocks of data and employs a 56-bit key for encryption, executing a total of 16 iterations to transform 64-bit plaintext into 64-bit ciphertext.

The AES is another prominent block cipher encryption algorithm, introduced by the National Institute of Standards and Technology (NIST) in 2000. As noted in various sources [5], [22], AES is a symmetric block cipher that utilizes the same key for both encryption and decryption processes. It was developed by Belgian cryptographers Daemen and Rijmen [23] and is notable for its flexibility in key lengths, supporting 128-bit, 192-bit, and 256-bit keys. Furthermore, AES is recognized as a highly effective block cipher, particularly well-suited for managing HDFS blocks.

3.2. AES and DES algorithms for data security

Our proposed encryption algorithm is designed to encrypt and decrypt files stored in the HDFS through a hybrid methodology that integrates two distinct algorithms: AES and DES. The AES algorithm operates in cipher block chaining mode, specifically the ECB mode, which is widely recognized as one of the most effective block cipher algorithms for managing HDFS blocks. Conversely, the DES algorithm is employed to encrypt the AES key after the file has been encrypted within HDFS. This key is subsequently decrypted using DES when the user intends to decrypt the file for executing a MapReduce job. In this framework, the encryption is performed using a 128-bit key.

To enhance the security of the Hadoop environment, the user will further encrypt this key utilizing the DES algorithm. When the user initiates a MapReduce job, the first step involves decrypting the key through a DES function. The detailed process of encryption and decryption for the proposed approach is outlined in the Pseudocode 1.

Pseudocode 1. Process for encrypting and decrypting files in HDFS

ENCRYPTION

```

Initialization
DES key ← random (64-bit)
AES key ← random (128-bit)
Block size ← 64MB
Encrypted key ← null
Ciphertext ← create empty file ( )
Encrypt HDFS (input file)
start
While (! end of file)
start
Block ← input file.read (block size)
if (length (block) =0)
break
else if (length (block) mod 16 ≠ 0)
block ← block + ' ' *(16 - length (block) mod16)
end if
Ciphertext ← ciphertext.write (AES encrypt (block, AES key))

```

```

end while
HDFS ← upload (ciphertext)
Encrypted key. ← ES encrypt (AES key, DES key)
    
```

DECRYPTION

```

Decrypt HDFS (ciphertext)
start
Decrypted key ← DES decrypt (encrypted key, DES key)
While (! end of file)
start
Block ← ciphertext .read (block size)
if (length (block) =0)
break
end if
Plaintext ← plaintext.write (AES decrypt (block, decrypted key))
end while
    
```

3.3. Encrypting files in HDFS

Initially, we assume that the user encrypts the file prior to its storage in HDFS. When the HDFS client initiates a request to write a file to HDFS [24], it invokes the create () method on the DFS, which in turn communicates with the NameNode to establish a new file within the filesystem's namespace. In the event of an error, the client encounters an IOException; if successful, the NameNode proceeds to create the file. Subsequently, the DFS provides an FSDataOutputStream. The client then begins the encryption process using the proposed method and writes the encrypted file into the FSDataOutputStream, which divides it into packets. These packets are directed to an internal queue known as the data queue, which is utilized by the data streamer. The data streamer's role is to request the NameNode to allocate new blocks while selecting a set of DataNodes for storing the replicas. The data streamer transmits the packets to the first DataNode, which retains the packet and forwards it to the subsequent DataNode, continuing this process until completion. Upon successful completion of the write operation, the DataNode sends an acknowledgment back to the HDFS client via DFS. Finally, the HDFS client concludes the process. The stages of this operation are illustrated in Figure 1.

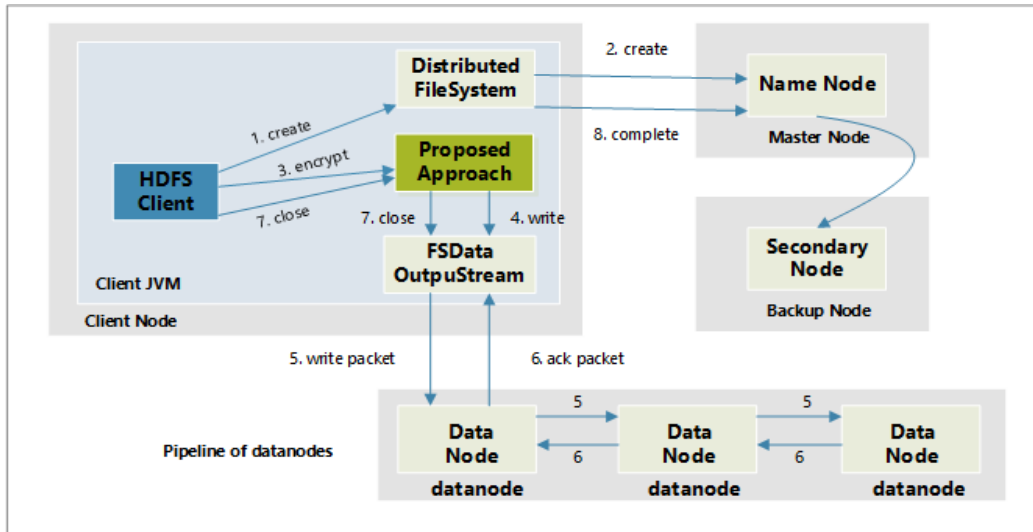


Figure 1. Writing a file with encryption in HDFS

3.4. Decrypting files in map task

The decryption process is initiated when the client opts to execute a MapReduce job, which is set up on the client node [25] and runs within a Java virtual machine (JVM). The JobClient transmits a new job request to the JobTracker, which subsequently returns a unique job identification number. Following this, the necessary execution files and distributed cache information are transferred to the relevant nodes. The job is then submitted, and the JobTracker utilizes the job ID to commence the job and verifies the input data required for execution. The JobTracker assigns the job to a task tracker that has the capacity to execute the map task. The task tracker then acquires the necessary resources to perform the task. Ultimately, the task

tracker initiates a new JVM and begins the decryption of the encrypted data using the proposed method, followed by the execution of the map task. The execution flow of the MapReduce job and the decryption process is illustrated in Figure 2.

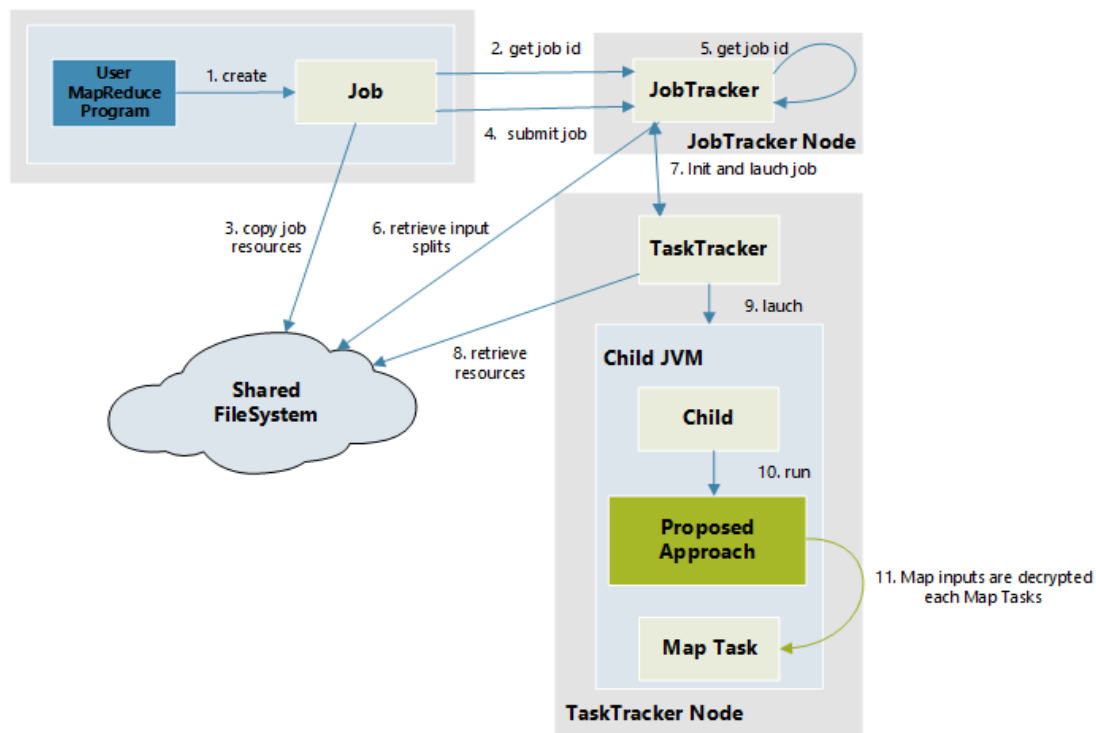


Figure 2. Decryption process at MapReduce job

4. EXPERIMENTAL ANALYSIS

4.1. Experimental environment

To assess the efficacy of our proposed methodology, we utilized Cloudera QuickStart 5.13.0 [26], an open distribution of Hadoop, which was installed and configured within an oracle VM VirtualBox 6.0 virtual machine. The Hadoop cluster established is a single node configuration, operating on version 2.6. This cluster comprises a single host, which is executed on a laptop equipped with an Intel Core i5-5200U processor, featuring 4 cores, a clock speed of 2.20 GHz, and 8 GB of RAM.

4.2. Results of the experiment

In this section, we provide a comprehensive analysis and results of our proposed technique in comparison to other established methods. Specifically, to evaluate the effectiveness of our algorithm, we conducted a comparison with the AES algorithm [11] and AES combined with OTP as referenced in [12]. Additionally, we utilized files of varying sizes ranging from megabytes to gigabytes (64 MB, 128 MB, 256 MB, 512 MB, and 1 GB).

The performance metrics of our technique were assessed in terms of encryption time, decryption time, file size, and throughput, which is defined as the size of the plaintext divided by the total time taken for encryption and decryption. For the implementation of the functions, we employed Python as the programming language. As detailed in Table 1, the time taken to write a 1 GB generic file to HDFS was 7.9913 minutes, while the encrypted file using the AES algorithm required 12.9751 minutes. In contrast, our proposed technique achieved encryption of a 1 GB file in HDFS in just 3.3483 minutes.

Furthermore, the decryption results presented in Table 2 indicate that our method also demonstrates a significant advantage. The decryption time for a 1 GB file using our technique was recorded at 12.1925 minutes, compared to 12.2115 minutes for AES with OTP and 14.0841 minutes for the AES algorithm. Although the differences in decryption times are less pronounced, our approach still provides superior performance, achieving an optimal balance between security and speed. This enhanced efficiency in both encryption and decryption underscores the robustness of our method for big data systems utilizing HDFS.

Table 1. File encryption performance comparison among generic HDFS, AES, AES with OTP and the proposed approach

File size (MB)	Generic HDFS (minutes)	AES algorithm (minutes)	AES and OTP (minutes)	Proposed approach (minutes)
64	0.4242	0.8704	0.7311	0.5762
128	0.8087	1.8216	1.3820	0.6136
256	1.6089	2.7396	2.5484	1.3447
512	3.0866	6.6682	4.8780	3.1175
1024	7.9913	12.9751	11.2511	3.3483
Total encryption time (minutes)	13.9197	25.0749	20.7906	9.0003
Throughput (MB/minutes)	142.5318	79.1229	95.4277	220.4370

Table 2. File decryption performance comparison among generic HDFS, AES, AES with OTP algorithms and the proposed approach

File size (MB)	Generic HDFS (minutes)	AES algorithm (minutes)	AES and OTP (minutes)	Proposed approach (minutes)
64	0.4242	1.3056	1.0950	0.6155
128	1.1137	2.1859	1.6560	1.3627
256	1.8642	2.8641	2.6554	2.2278
512	4.2917	8.9494	6.5361	5.0441
1024	11.2232	14.0841	12.2115	12.1925
Total decryption time (minutes)	18.9170	29.3891	24.1540	21.4426
Throughput (MB/minutes)	104.8792	67.5080	82.1396	92.5260

As indicated in Table 3, the generic file size of 1 GB increases to 1.5 GB when encrypted with AES, while it decreases to 1.2 GB when both AES and OTP are applied. In contrast, our method maintains the encrypted file size at the original generic file size. In order to effectively illustrate this comparison, we created graphical representations for each criterion. Figure 3 depicts the estimated time required for the encryption process across various algorithms, including the proposed method. Additionally, Figure 4 illustrates the time required for the decryption process when employing various algorithms alongside the proposed method.

Table 3. File size comparison among generic HDFS, AES, AES with OTP algorithms and the proposed approach

File size (MB)	Generic HDFS (MB)	AES algorithm (MB)	AES and OTP (MB)	Proposed approach (MB)
64	64	96.0	74.7	64
128	128	192.0	149.3	128
256	256	384.0	298.7	256
512	512	768.0	597.3	512
1024	1024	1536	1228.8	1024

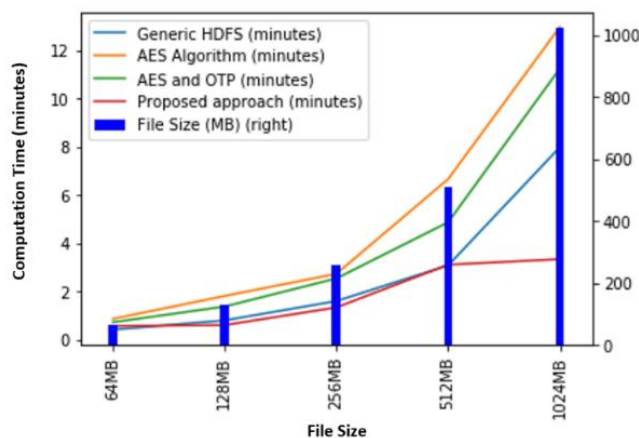


Figure 3. Encryption time (minutes) of generic HDFS, AES, AES with OTP and proposed approach

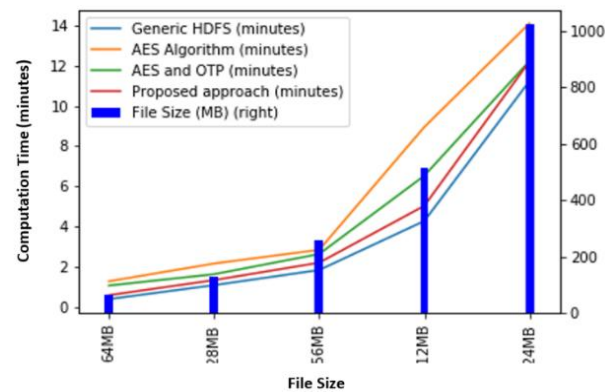


Figure 4. Decryption and job execution time (minutes) of generic HDFS, AES, AES and OTP and proposed approach

4.3. Discussion

In this research, we assessed the efficacy of a hybrid encryption strategy that integrates the AES and DES algorithms for the protection of HDFS files. Our findings indicate a notable decrease in encryption duration when compared to conventional techniques, with our method requiring 3.3483 minutes to encrypt a 1 GB file, in contrast to 12.9751 minutes for AES alone. This enhanced efficiency can be attributed to the synergistic benefits of both algorithms, which not only bolster performance but also ensure robust security.

When juxtaposing our findings with those from other investigations, such as [22], which focused on AES in isolation and AES combined with OTP, it becomes evident that our method provides superior encryption speeds and minimizes file size. Furthermore, our approach preserves the encrypted file size to be equivalent to that of the original, unlike the AES and OTP combination, which tends to inflate the encrypted file size. The proposed scheme demonstrates a higher throughput relative to both the AES and AES with OTP methodologies. It is well-established that increased throughput correlates with reduced energy consumption, indicating that the energy usage of the AES and DES combination is lower than that of the AES and AES with OTP configurations.

Nonetheless, a limitation of our study is the reliance on a single-node Hadoop cluster, which may not accurately reflect performance in a more extensive and distributed setting. Interestingly, we noted a consistent enhancement in throughput across various file sizes, hinting at potential scalability benefits. The primary aim of this research was to improve data security within Hadoop's HDFS through the implementation of a hybrid encryption strategy. The findings underscore the necessity of refining encryption methodologies to adequately protect extensive datasets. This investigation lays the groundwork for additional studies on hybrid encryption techniques, indicating that future inquiries should focus on assessing performance in multi-node clusters and practical applications. Tackling these aspects will be essential for formulating more robust solutions for big data security.

5. CONCLUSION

In this research, the authors focused on enhancing data security within the HDFS. To address the security challenges associated with big data, a hybrid methodology was proposed that integrates both AES and DES encryption techniques. Each file is encrypted prior to being stored in HDFS, utilizing the AES and DES algorithms, and is subsequently decrypted during the map task. To assess the effectiveness of the proposed method, files of varying sizes were utilized in the evaluation. The experimental results indicated that this approach is more time-efficient compared to other methods implemented. Additionally, the proposed technique demonstrated superior throughput while maintaining lower power consumption. Importantly, this method does not result in an increase in file size.




REFERENCES

- [1] B. Saraladevi, N. Pazhaniraja, P. V. Paul, M.S. S. Basha and P. Dhavachelvan, "Big data and Hadoop-a study in security perspective," *Procedia Computer Science*, vol. 50, pp. 596-601, 2015, doi: 10.1016/j.procs.2015.04.091.
- [2] T. White, "Hadoop: the definitive guide". Sebastopol, CA, USA: O'Reilly Media, 2015.
- [3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large cluster," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008, doi: 10.1145/1327452.1327492.
- [4] S. Ghemawat, H. Gobiuff, and S. T-Leung, "The Google file system," *ACM SIGOPS Operating Systems*, vol. 37, no. 5, pp.29-43, 2003, doi: 10.1145/1165389.945450.




- [5] I. O'Malley, K. Zhang, S. Radia, R. Marti, and C. Harrell, "Hadoop security design," Yahoo! Technical Report, 2009. [Online]. Available: <https://issues.apache.org/jira/secure/attachment/12428744/HadoopSecurityDesign.pdf>. (Accessed: Jan. 29, 2024)
- [6] Ishwarappa and J. Anuradha, "A brief introduction on big data 5Vs characteristics and Hadoop technology," *Procedia Computer Science*, vol. 48, pp. 319-324, 2015, doi: 10.1016/j.procs.2015.04.188.
- [7] H. J. Hadi, A. H. Shnain, S. Hadishaheed and A. H. Ahmad, "Big Data and five V'S characteristics," *International Journal of Advances in Electronics and Computer Science*, vol. 2, no. 1, 2015.
- [8] A. Murthy, V. K. Vavilapalli, D. Eadline, J. Niemiec, and J. Markham, "Apache Hadoop YARN: moving beyond MapReduce and batch processing with Apache Hadoop 2," Boston, MA, USA: Addison-Wesley, 2014.
- [9] M. R. Ghazi, and D. Gangodkar "Hadoop, MapReduce and HDFS: a developers perspective", *Procedia Computer Science*, vol. 48, pp 45-50, 2015, doi: 10.1016/j.procs.2015.04.108.
- [10] R. P. Padhy "Big data processing with Hadoop-MapReduce in cloud systems," *International Journal of Cloud Computing and Services Science*, vol.2, no.1, 2012, doi: 10.11591/closer.v2i1.1508.
- [11] S. Park and Y. Lee, "Secure Hadoop with encrypted HDFS," in *International Conference on Grid and Pervasive Computing*, 2013, pp. 134–141, doi: 10.1007/978-3-642-38027-3_14.
- [12] H. Mahmoud, A. Hegazy, and M. 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)*, Feb. 2018, vol. 2018-March, pp. 109–114, doi: 10.1109/ITCE.2018.8316608.
- [13] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques," *International journal on computer science and engineering*, vol. 4, no. 5, pp. 877–882, 2012.
- [14] M. R. Jam, L. M. Khanli, M. S. Javan and M. K. Akbari, "A survey on security of Hadoop," *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2014, pp. 716-721, doi: 10.1109/ICCKE.2014.6993455.
- [15] H. Zhou and Q. Wen, "Data security accessing for HDFS based on attribute-group in cloud computing," *Proceedings of the International Conference on Logistics, Engineering, Management and Computer Science*, 2014, doi: 10.2991/lemcs-14.2014.255.
- [16] H. -Y. Lin, S. -T. Shen, W. -G. Tzeng and B. -S. P. Lin, "Toward data confidentiality via integrating hybrid encryption schemes and Hadoop distributed file system," *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan, 2012, pp. 740-747, doi: 10.1109/AINA.2012.28.
- [17] C. Yang, W. Lin, and M. Liu, "A novel triple encryption scheme for Hadoop-based cloud data security," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, Sep. 2013, pp. 437–442, doi: 10.1109/EIDWT.2013.80.
- [18] Y. Tian, "Towards the development of best data security for big data," *Communications and Network*, vol. 09, no. 04, pp. 291–301, 2017, doi: 10.4236/cn.2017.94020.
- [19] M. I. Maheswari, S. Revathy, R. Tamilarasi, "Secure data transmission for multi sharing in big data storage," *Indian Journal of Science and Technology*, vol. 9, no. 21, pp. 1-9, 2016. doi: 10.17485/ijst/2016/v9i21/95164.
- [20] I. Sumartono and A. P. U. Siahaan, "Encryption of DES algorithm in information security," *International Journal for Innovative Research in Multidisciplinary Field*, vol. 4, no. 10, 2018.
- [21] Kiramat, "Comparison of various encryption algorithms for securing data," doi:10.31224/osf.io/xzv56.
- [22] A. M. Abdullah, "Advanced encryption standard (AES) algorithm to encrypt and decrypt data," 2017. [Online]. Available: <https://www.example.com/advanced-encryption-standard>. (Accessed: Apr. 12, 2023)
- [23] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—the advanced encryption standard*. Berlin, Germany: Springer-Verlag, 2002.
- [24] H. Khizou, "Big data from B to A: the Hadoop distributed filesystem-HDFS, *Towards Data Science*, 2019.
- [25] M. Barreto, S. Nesmachnow, A. Tcherykh, "Hybrid algorithms for 3-SAT optimisation using MapReduce on clouds". *International Journal of Innovative Computing and Applications*, 2017, doi: 10.1504/IJICA.2018.10011774.
- [26] Cloudera, Inc., "Cloudera CDH 5.13.0 downloads," [Online]. Available: <https://www.cloudera.com/downloads/cdh/5-13-0.html>. Accessed: Dec. 29, 2023.

BIOGRAPHIES OF AUTHORS






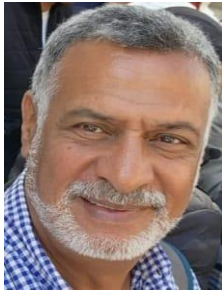
Fayçal Zine-Dine    is a Ph.D. candidate researcher at the Faculty of Sciences, Chouaib Doukkali University. He holds a master's degree in information systems engineering from the same faculty, obtained in 2017. He received his bachelor's degree in software engineering from the Faculty of Sciences, Moulay Ismail University in Meknes in 2014. He is currently pursuing his research at the ELITES laboratory of the Faculty of Sciences, Chouaib Doukkali University. His research interests encompass big data, information systems security, artificial intelligence, and business intelligence. He can be contacted at email: zdfaycal@gmail.com.






Sara Alcabnani    was born in 1992 in Safi, Morocco. She obtained a master's degree in information systems engineering from the Faculty of Science at Chouaib Doukkali University in 2018. She obtained a bachelor's degree in computer development from the Polydisciplinary Faculty of Safi, Cadi Ayyad University in 2015. Recently, she obtained her doctorate in computer science and continues her research in the ELITES Laboratory at the Faculty of Science, Chouaib Doukkali University. Her current research focuses on machine learning, sentiment analysis, social networks, social media mining, e-reputation, business intelligence and decision-making systems. She currently works as a senior IT executive at Morocco's Caisse Nationale de Sécurité Sociale (CNSS). She can be contacted at email: s.alcabnani@ucd.ac.ma or saraalcabnani@gmail.com.



Prof. Ahmed Azouaoui    received his license in June 2001 in computer science and engineering and a master's degree from University of Mohammed V, Rabat, Morocco in computer science and telecommunication in 2003. He obtained his Ph.D. in 2014 at the Department of Computer Science of the National School of Computer Science and Systems Analysis, Rabat, Morocco, in computer science and engineering. Currently, he is a professor at Higher School of Technology, Ibn Tofail University, Morocco. His domains of interest are artificial intelligence, coding theory, and information systems. He can be contacted at email: a.azouaoui@uit.ac.ma.



Prof. Jamal El Kafi    is a seasoned academic with a Ph.D. in robotics and certifications in quality auditing and coaching. As a full professor at Chouaib Doukkali University, he has established himself as a leading expert in artificial intelligence, ICT, and educational technologies. He heads the DIS research team, focusing on decision support systems, and is an associate member of the ELITES Laboratory. He is the founder and president of TENORS, an association dedicated to promoting new technologies and scientific research. With extensive experience in AI, IS, quality management and international education systems, he has authored numerous scientific articles, evaluated research papers, and supervised multiple Ph.D. theses. He can be contacted at email: jelkafi@gmail.com.