Hybrid Encryption Algorithm for Information Security in Hadoop

Youness Filaly[1], Fatna El mendili[2], Nisrine Berros[3] Younes El Bouzekri EL idrissi[4]
Engineering Sciences Laboratory, Ibn Tofail University, National School of Applied Sciences, Kenitra, Morocco[1,3,4]
Image Laboratory, Moulay Ismail University of Meknes, School of Technology, Meknes, Morocco[2]

Abstract—**Network security has gained importance in recent years. Information system security is greatly aided by the development of encryption as a solution. To safeguard the shared information, several strategies are required. Thanks to the cutting-edge Internet, networking corporations, health information, and cloud applications, our data is growing exponentially every minute. In order to handle enormous amounts of data efficiently, a new application called Hadoop distributed file system (HDFS) was created. However, HDFS doesn't come with any built-in data encryption tools, which poses serious security risks. In order to increase data security, encryption techniques are established; nevertheless, standard algorithms fall short when dealing with bigger files. In this study, huge data will be secured using a novel hybrid encryption algorithm that combines CP-ABE (encryption based on the features of the encryption policy), AES (advanced standard encryption), and RSA (Rivest-Shamir-Adleman). The suggested model's performance is compared against that of traditional encryption algorithms like DES, 3DES, and Blowfish in order to demonstrate improved performance as it relates to decryption time, encryption time, and throughput. The results of the studies demonstrate that our suggested method's algorithm is more secure.**

Keywords*—*Hadoop distributed file system (HDFS); big data security; data encryption; data decryption

I. Introduction

Megadata is typically believed to be a collection of data whose structure is too huge or too diverse and complicated to be handled by standard data processing tools [1]. The problems of Big data include acquiring, storing, analyzing, transporting, sharing and displaying the information it contains [1]. Scientists, entrepreneurs and healthcare professionals are frequently needed to utilize data from a range of sources, including megadata from worldwide literature, the Internet, medical records, patient registries and even "smart" gadgets [1]. Smart phones increased use in recent years has resulted in a sharp rise in the amount of data created by social networking services (SNS). Data including multimedia material in a variety of formats has expanded to big data scale. Big data is a contemporary industrial research hotspot. Big data started to take off and get more and more attention as the cloud age came into being. Numerous apps, sensor networks, social networking sites and enterprises big and small handle this massive volume of data [2], [3]. Big Data challenges may be described in a number of ways, not least by reference to the 4 Vs [4]: volume, velocity, variety and veracity. The work of living with large data is different for every situation.

- Volume: Data with a size of many terabytes or petabytes.
- Variety: There are several types of data. organized, semiorganized, and unorganized.
- Velocity is an abstraction layer that allows Big Data systems to store data independent of the incoming or departing flow by specifying the different speeds at which data streams might enter or exit the system.
- Veracity: The information is under question. relates to the data's credibility, taking into account data availability, confidentiality, and integrity. Companies must guarantee that the data are accurate, as well as any analysis that are done on the data.

Using a distributed, trustworthy, and scalable computing infrastructure, big data sets may be handled and stored using the open-source Hadoop framework [5]. Because of Hadoop's cheap cost, quick processing, fault tolerance, and flexibility, large clusters or public cloud services often employ it. Hadoop [6] is an open source Java-based distributed computing framework consisting of two modules: MapReduce and Hadoop Distributed File System. It is used as a framework for cloud storage (HDFS). By simply defining the map and reduce functions, users can process large amounts of data using MapReduce, enabling them to efficiently use thousands of commodity computers in parallel [7], [8], [9]. HDFS is used to store data on distributed clusters of machines. Large clusters or public cloud services such as Yahoo!, Facebook, Twitter and Amazon are the places where Hadoop is most often used [10]. The popularity of these applications has shown Hadoop's scalability, but it lacks security for data storage by design. Hadoop has weaknesses in its security features even if its processing capacity is far more than that of traditional data processing systems. As the Hadoop design is not meant to be safe, it offers no security mechanisms to protect data while it is being stored or transferred. Currently, corporations are analyzing consumer information and location data obtained in numerous areas and utilizing it for marketing activities. As a consequence, sensitive personal data may be revealed when consumer data is evaluated. There would be significant reputational harm and legal implications as a result of the data breach. In addition, businesses store and process a large amount of data, all of which has to be protected in HDFS storage using encryption, threat detection, and logging systems. These methods let systems quickly detect vulnerabilities and protect user data. To safeguard data from the generating phase through the storage phase, several solutions have developed recently. Using data fabrication methods and limiting access during data production improves data privacy. Data security and privacy are mostly ensured during the storage phase via encryption methods [11].

The structure of simple file authorization and access control techniques constitutes the security service of the Hadoop project. To protect HDFS files stored in data nodes and to move files between data nodes when performing MapReduce operations, encryption is the best option. The process of converting plain text into ciphertext is known as encryption. By allowing users to be properly authenticated and prohibiting others, the transformation of explicable data into an incomprehensible form protects data confidentiality [12]. Data confidentiality and integrity are two objectives that can be achieved in Hadoop when using encryption. There are two different types of cryptographic keys: symmetric key cryptography, sometimes called secret key cryptography. Asymmetric key cryptography, sometimes called public key cryptography [13]. Stream ciphers, such as RC4 and OTP, and block ciphers, such as the AES, DES, 3DES and BLOWFISH algorithms, are generally used in secret key cryptography methods [14]. Using encryption techniques, several researches [15], [16], [17] indicated that the file size was 1.5 times larger than the original file and that download time also increased.

A. The Scope of this Paper

In order to protect data on Hadoop, the main objective of this research is to solve the problem of data security in Hadoop. This was solved by recommending the implementation of a new strategy that combines HDFS files with the CP-ABE (attribute-based encryption) technique with RSA (Rivest-Shamir-Adleman) and AES (Advanced Encryption Standard) algorithms to speed up the upload and download of the encrypted file. The main contributions of this paper are summarized below:

- We present a thorough related works on encrypting data in HDFS.
- We describe the architecture of big data encryption system.
- We present a hybrid encryption technique in HDFS for Big Data security.

In the subsequent sections, we present a structured outline of the remaining content. Section 2 delves into a brief literature review, examining key studies and their findings in the field. Section 3 introduces our suggested hybrid encryption approach, providing insights into its design and implementation. The experimental results obtained from applying this approach on a substantial dataset are presented in Section 4, along with a thorough analysis. Finally, Section 5 concludes the paper by summarizing the main findings and discussing potential avenues for future research.

II. Related Works

This section includes short appraisals of the literature on Big Data security and encryption approaches. The security and privacy aspects of Big Data applications are considerably strengthened by researchers utilizing a range of encryption technologies [18], [19], [20]. A privacy-preserving auction mechanism is utilized in the homomorphic cryptography and secure network protocol architecture proposed by W. Gao et al. [21] to increase user and third-party service provider confidence and data privacy. Data communication between the user and third-party service providers is safeguarded by the homomorphic encryption technique. For better data security, the revised approach leverages signature-based verification. However, as the user base and file size rise, system performance declines.

The completely homomorphic encryption system presented by A. Alabdullatif et al. [22] avoids dangers and data privacy breaches in the Big Data environment from both inside and outside. The resilient cryptosystem splits computation and data into two different portions following task analysis. The system's capacity to digest data is substantially sped by this technique, which also achieves a high degree of accuracy. C. Xiao et al. [23] proposes an accelerated approach for solving a range of complicated data encryption and computing challenges. A secure data storage system with adaptive cryptographic acceleration that dynamically enhances the working modes of huge data files is given for big data encryption. Compared with comparable software and hardware gas pedals, the design work achieves a better compromise.

By the use of an encryption technique, the data analysis model described by K. Sharma et al. [24] effectively addresses the privacy concerns in the interchange of health information. The patient-centric data access control mode addresses the privacy concerns with regard to health information and the need for data encryption. The patient file is encrypted and sent utilizing a variety of domains when employing the suggested RSA-based encryption. Even if the encryption paradigm successfully addresses the security and privacy concerns, data transmission across many domains is required. Comparatively to other data transmission techniques, this raises the system cost.

The results in [25] describe 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 assaults. Data secrecy in huge data streams is guaranteed by the selective encryption technique described by D. Puthal et al. [26]. Data integrity and confidentiality are security factors that affect how reliable the obtained data is. To increase the efficiency of encrypting and decrypting data streams while retaining data integrity and privacy, the authors employ the selective encryption technique.

The challenges connected with employing standard cryptographic techniques have been overcome in the attribute-based encryption stated by P. Perazzo et al. [27]. The encryption paradigm addresses the need for accurate access control in a vast data environment. Flexible rules improve access control to encrypted data while simplifying the management of large data volumes. Ten unique criteria are utilized to measure performance, and cryptographic acceleration is employed to boost performance. The key benefits of this encryption system are its minimal memory and energy needs. The shortcoming of standard attribute-based encryption (ABE) approaches is overcome by the hybrid attribute-based encryption (ABE) model described by H. Deng et al. [28]. The hybrid model provided adds a new proxy encryption to turn ABE ciphertext into IDE (identity-based encryption) ciphertext, since the rules stated by standard models become outdated after a specific length of time. Data collisions are prevented and security is promoted by using identity-based encryption and key randomization properly.

Using the CP-ABE approach, P.S. Challagidad et al. [29] investigated the difficulties of unauthorized data access and confidentiality concerns in enormous data clouds. The encryption technology addressed the demand for multi-authority access control and data secrecy. Users get precise data and access thanks to the role hierarchy algorithm and hierarchical access structure. The two key benefits of hierarchy algorithms are computation speed and minimum storage needs.

Attack detection and intrusion detection are crucial concepts to take into account while researching Big Data security due to their impact on data privacy and confidentiality. To boost data security, several attack detection models and intrusion detection methodologies have been created. Using instruction sequences, a two-stage attack detection system presented by S. Aditham et al. [30] defends communication protocols. To assess system needs, these instruction sequences are further mapped to nodes. The encryption and decryption method presented by J. S. Raj et al. [31] takes into consideration the shortcomings of attribute-based encryption matching techniques, which impair the performance of the encryption system. For outsourced data operations, a lightweight, fine-grained data sharing strategy is employed to bypass this challenge. This increases overall data security and avoids the leakage of decryption keys.

Abd al wahid, S. M. J. et al. [32] suggested a solution for encrypting all files stored in HDFS utilizing public-key cryptography to safeguard them all. Acquired data is encrypted in HDFS during the data collecting process using the suggested data encryption technique (Rabin RZ). The suggested technique is contrasted with the Paillier method and the default Rivest-Shamir-Adleman (RSA) cryptosystem, respectively. Compared with previous cryptosystems, the suggested technique provides more powerful computational complexity and lower latency than the alternatives.

In order to strengthen transaction security against unauthorized access and to verify the speedy data transaction with minimal encryption and decryption time, Motupalli, R. K. et al. [33] presented an effective mixed algorithm design utilizing the Salsa20 and AES algorithm. The impressive throughput achieved in this hybrid framework shows how effective the recommended algorithmic structure is on current platforms.

III. Big Data Encryption System Architecture

With the user interface the system offers, the client may communicate with the large data storage system. The major functionalities of the user interface include identity authentication, which provides users with login authentication and operation authority authentication, big data management, which offers authorized users services like browsing and replicating huge data, and other interfaces. Storage cluster and metadata cluster are the two primary components of the big data storage system. Storage cluster is used to store user files and other non-metadata data, whilst metadata cluster is used to store metadata like user and file information. The hierarchical structure of the system is analogous to the large data storage system as a whole, which consists of four layers: the access layer, the application interface layer, the data management layer and the storage layer. Users may access cloud storage systems via sites like login and data operations, which are part of the direct user-system interface provided by the access layer. To access HDFS, HBase, and MySQL, the application interface layer offers web services and APIs. In response to user requests, the application server may call the appropriate routines to perform a particular data activity or user identity authentication. The data storage systems HDFS, HBase, and MySQL that can perform the operations of adding, removing, editing, and verifying data are included in the management layer. The physical storage devices that make up the bulk of the storage layer are virtualized into a single entity that offers storage services to the outside world. It also performs status monitoring and centralized management of storage resources concurrently. The system's basic four modules for module and function design are user file systems, file sharing, user information management, and personnel management. File browsing, file uploading, download, sharing, file inquiry, file management, shared file browsing, shared file downloading and retrieving, removing sharing files, changing passwords, adding users, deleting users, resetting passwords, etc. are some of the specific features.

Fig. 1 depicts the system's general design.

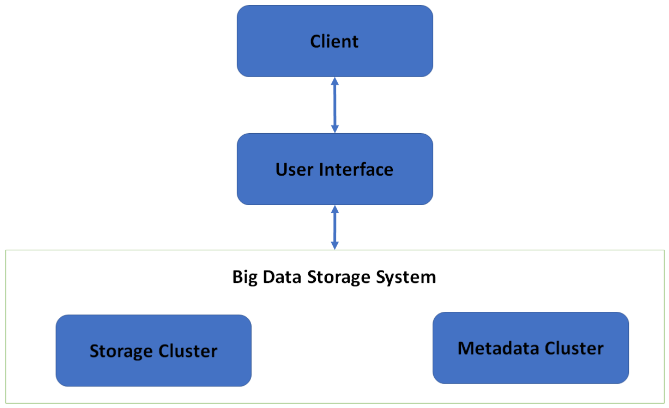

Fig. 1. Overall structure of system.

IV. Proposed Work

A. Overview

One potential strategy to combine HDFS files with CP-ABE, RSA, and AES algorithms to speed up uploading and downloading while minimizing the size of the encrypted file could involve the following steps:

- Partitioning data into smaller chunks: Let's assume that we have a large file of size L that we want to upload or download. Instead of uploading or downloading the entire file at once, we can partition the file into n smaller chunks, each of size L/n. By doing this, we can distribute the load across multiple machines, which can significantly speed up the process. Mathematically, we can represent this as:

$$\mathbf{L = n(L/N)}$$

- The hybrid CP-ABE, AES, and RSA encryption scheme is a secure data communication approach that utilizes three encryption methods to provide confidentiality, integrity, and access control. The scheme involves the use of a cyclic group G, a hash function H, and a bilinear pairing e: $GxG \rightarrow GT$ for cpABE encryption. The master key mk and user secret keys sk_i are generated using a set of equations and algorithms. The mk consists of (s, t, T) where s is a random element of Zp, t is a random element of Zp, and T is a bilinear map from G to GT. The user secret key sk_i consists of (D_i, T^i) where D_i is an element of G and T^i is T raised to the power of the attribute vector of i. To encrypt a message M, a random symmetric key K is generated, and the message is encrypted using AES. The symmetric key K is then encrypted using RSA with the recipient's public key, and both ciphertexts are encrypted using cpABE with a given access policy ω. To decrypt the message, the recipient uses their RSA private key to decrypt the encrypted symmetric key K, and the message is decrypted using the decrypted symmetric key. The security of the proposed scheme is analyzed using formal security definitions and proofs. The performance evaluation of the scheme is conducted in terms of computation time and communication overhead, and the results demonstrate that the scheme is efficient and practical for real-world applications.

For more Mathematical explanation in the hybrid cpABE, AES, and RSA encryption scheme we found four step:

- Setup:
  - Let p and q be two large prime numbers. The product of these primes, n = pq, is used as the modulus for RSA encryption:
    $$n = pq$$
  - The totient of n, $\phi = (p-1)(q-1)$, is used to compute the RSA public and private exponents, e and d, respectively:
    $$\phi = (p-1)(q-1)$$
    e, d are such that e*d $\equiv$ 1 (mod ϕ) and 1 ¡ e ¡ ϕ, where e is the public exponent and d is the private exponent.
  - A cyclic group G is chosen with order n, and a generator g is selected from this group. This group is used for cpABE encryption:
    $$G = g^x \pmod{n} : x \in Z\_n^*$$
  - A hash function H is chosen that maps a bit string of arbitrary length to an element of the group G:
    $$H : 0, 1^* \rightarrow G$$
- Key Generation:
  - The authority generates a master key (MK) consisting of (p, q, n, ϕ, e, d, G, g, H).
  - For each user i, the authority generates a secret key SKi consisting of a set of attributes Ai and a private key si. These secret keys allow users to decrypt messages that are encrypted with CP-ABE.
- Encryption:
  - To encrypt a message M for a set of attributes S, the encryptor first generates a random symmetric key K that will be used to encrypt the message with AES:
    $$K = Random()$$
  - The encryptor encrypts the message M using AES with the symmetric key K, producing the ciphertext C1:
    $$C1 = AES\_Encrypt(M, K)$$
  - The encryptor encrypts the symmetric key K using RSA with the public key (n, e), producing the ciphertext C2:
    $$C2 = RSA\_Encrypt(K, (n, e))$$
  - The encryptor encrypts both C1 and C2 using cpABE with the policy P(S), resulting in the final ciphertext C:
    $$C = cpABE\_Encrypt(P(S), (C1, C2))$$
- Decryption:
  - To decrypt the ciphertext C for a user i with attributes Ai, the user first decrypts C2 using their private key si to obtain the symmetric key K:
    $$K = RSA\_Decrypt(C2, si)$$
  - The user then decrypts C1 using the symmetric key K to obtain the plaintext M:
    $$M = AES\_Decrypt(C1, K)$$

The triple encryption approach provides better data protection when compared to traditional encryption techniques. ABE has recently attracted a lot of attention because of its decentralized access control and secure communication skills in dynamic environments. However, user-defined rules or processes cannot define the encryption process. To provide users more influence over the encryption process, access control rules are specified as ciphertext policies. Along with setting the properties, users may also control the encryption and decryption policies. These access control restrictions also provide cryptographical protection for data during transmission and storage. Only the properties are encrypted in CP-ABE; the whole block is not.

The hybrid encryption algorithms model for encrypting /decrypting files is seen in Fig. 2. The selection of input file properties is where the operation starts. Qualities are picked utilizing logical combinations and user preferences. Once the criteria have been determined, a set of rules is built expressly for these traits and encryption is executed. The AES algorithm creates a key that is used to safeguard the file after it has been encrypted for two-level security. Then, to safeguard the encrypted file, the AES key produced using the RSA process is applied. RSA AES key encryption is used to offer authentication while decrypting the encrypted file. If they do, it goes to the next stage; if not, the decryption process pauses and records the endeavor as an intrusion. Once the characteristics of the cipher match those of the key, if the key is the same as the actual key, attributes are applied and the file is then decrypted. If not, it is likewise categorized as an incursion at this level. The final file that has been encrypted will be plain text and useable in the appropriate program.

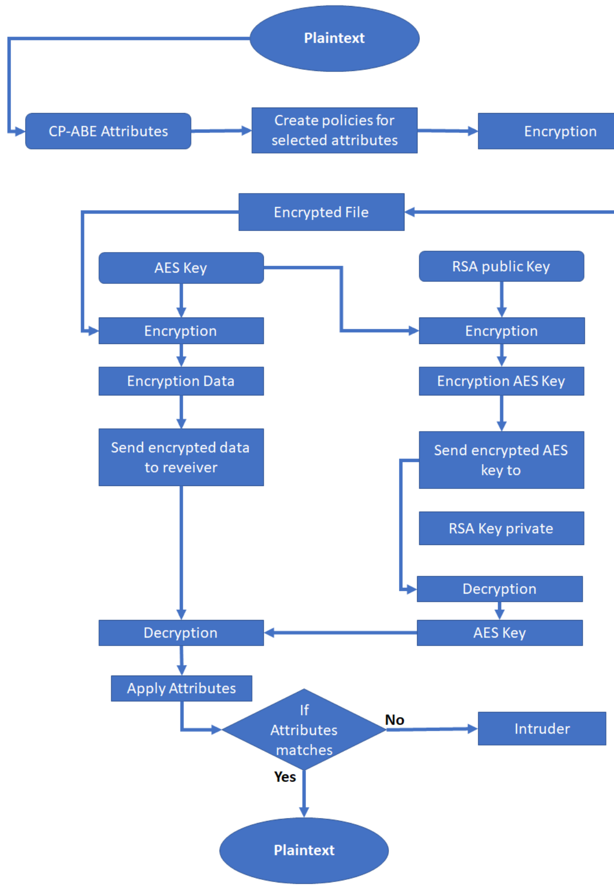

Fig. 2. Overview of the proposed hybrid encryption model.

B. Encrypting Files in HDFS

When submitting files to the HDFS system, the encryption operation is carried out. When files are successfully encrypted, data security is increased. Fig. 3 depicts the HDFS system's encryption process.

The actions taken during the encryption process are listed below in brief:

Step 1 : The system of distributed files is utilized in the first stage to simplify interaction between the HDFS user and the master node.

Step 2: The system of distributed files forwards the request containing the demand to create a new file to the master node.

Step 3: The master node analyzes the data node's availability of space and picks the right data node.

Step 4: Data node information is exchanged with the distributed file system and subsequently transmitted to the HDFS client.

Step 5: Before encrypting the file, the client transmits the attributes . The file is encrypted in the data node when the characteristics are specified.

Step 6: A key is generated using the AES approach

Step 7: The AES key generated encrypted using RSA Algorithm and applied to the encrypted file in order to safeguard it.

Step 8: Using output data streams from a distributed file system, a writing process begins from a client to a particular data node.

Step 9: Data from the current data node is relocated to another data node if the write operation is complete.

Step 10: The master node stores information about the current data node and replication data node throughout the replication operation.

Step 11: Using the distributed file system, an acknowledgment is sent to the HDFS client once the data has been correctly duplicated on the secondary data node.

Step 12: After receiving the acknowledgment, the HDFS client pauses the writing process.

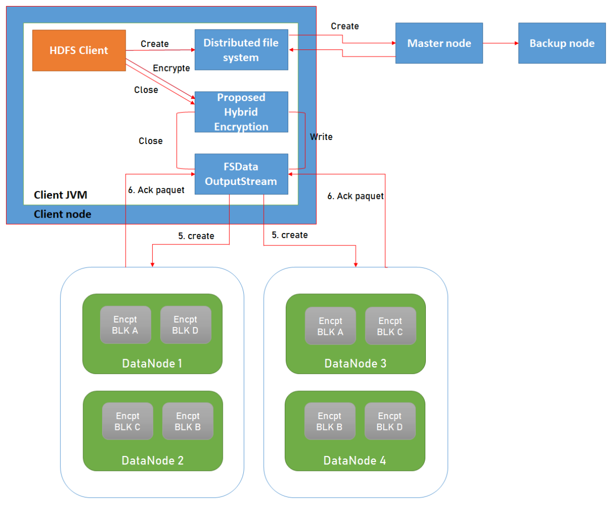

Fig. 3. Encryption process in HDFS.

C. Decryption Files in HDFS

When reading the files from the HDFS system, the decryption phase is carried out. Before reading the operation, the file must be decrypted, and using this technique may assist identify any illegal access or intruders. Fig. 4 depicts the HDFS system's decryption process.

The activities followed during the decryption method are explained as follows:

Step 1: The HDFS client talks with the master node across the distributed file system to begin the decryption process.

Step 2: A distributed file system sends a request comprising a request to read a file to the master node.

Step 3: The master node gives info about the data node that holds the encrypted files.

Step 4: The HDFS client begins the operation by picking data from the chosen block using the file system data input stream.

In Step 5: The client inputs the attributes to decrypt the file if the password provided for authentication matches.

Step 6: Access is regarded as an invasion or illegal access if the matching procedure is failed.

Step 7: After getting the acknowledgment, the HDFS client pauses the reading operation.

Step 8: After getting the HDFS client acknowledgment, the reading process is fully complete.

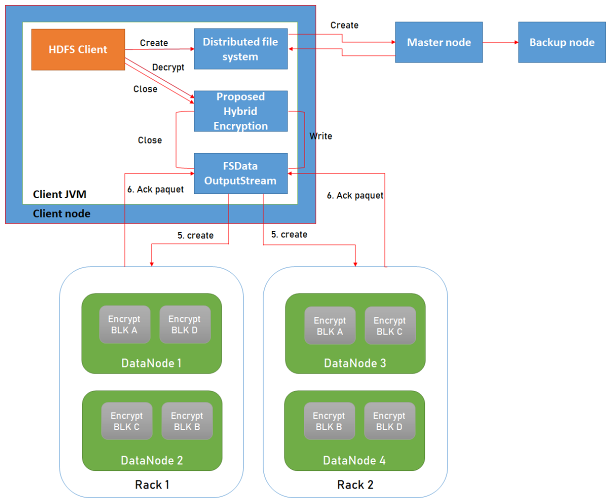

Fig. 4. Decryption process in HDFS.

D. Proposed Approach Pseudo Code

```
Initialization
Data: Plaintext to encrypt
Result: Encrypted cipher-text
Begin Algorithm
  1: Initialize characteristics
  2: Create a set of guidelines based on the characteristics
     and logical pairings
  3: Apply characteristics to the file and use a set of rules to
     encrypt it.
  4: Using the AES-generated key encryted by RSA, secure
     the file. =0
```

```
Decryption
Begin Algorithm
  1: begin the authentication process by matching AES keys
     =0
if user input = RSA Key AND user input = AES Key
  then
    Allow user to process next step ;
else
    Declare a threat;
end
Perform characteristics matching;
if characteristics = characteristics then
    Decrypt encryted file;
else
    Declare a threat;
end
```

V. Experiments and Results

The suggested hybrid encryption technique for large data security in the HDFS environment is confirmed by tests done in the Hadoop system installed in the CPU Intel® core i5 2.40GHz, 8 processors, 16 GB memory, 128G Solid state drive and operating system CentOS release 7.5.1804. The master node is chosen as one of the nodes, while the data nodes are the other nodes. Performance evaluation of encryption and decryption is done on files of various sizes. Throughput, encryption and decryption times, efficiency, and other factors are compared to the conventional DES, 3DES, and Blowfish algorithms. We also contrasted it with a different hybrid approach, that consists of AES and OTP algorithms. The parameters used in the proposed work on five CVs files with different sizes (64 MB, 128 MB, 256 MB, 512 MB , 1024 MB) depicted in Table I.

TABLE I. Parameters

Simulation No.	Parameter	Range/Value
1	Input file size	64 MB to 1024 MB
2	Memory	16
3	Key length	128, 256 bit

A. Data Security

Five algorithms are employed in this experiment to encrypt and decode the same data. The results show that the best algorithm is our hybrid approach, which provides the highest reliability and security for the data sent when the DES, 3DES , Blowfish hybrid algorithm (OTP and AES) and our approach algorithms are compared, the differences in encryption and decryption times evaluated in Fig. 5. The encryption-decryption timing for the five is provided in Tables III and IV with varying file sizes. Table V displays the entire processing time in minutes. From the findings shown in Table V, it's evident that our technique is superior than the other algorithms (DES 3DES, Blowfish and Hybrid) after evaluating all five algorithms and calculating the time required to finish processing on many files of varying sizes. Our hybrid approach encryption method gives the best degree of data security of the other algorithms, and its performance is substantially quicker than the DES,3DES, Blowdish and hybrid algorithms. Comparison of Encryption Algorithm is given in Table II.

Calculating how long it takes to convert plain text into ciphertext gives us the encryption time. The Table III shows that the suggested model takes the fewest amount of time for each file. A maximum file size of 1 GB of data takes around 5 minutes to encrypt, compared to 13 minutes for DES, 12.5 minutes for 3DES, 11.8 minutes for Blowfish, and 6.2 minutes for hybrid. The encryption time grows progressively as the file size increases.

TABLE II. Comparison of Encryption Algorithm

Factors	**AES**	**DES**	**3DES**	**RSA**	**Blowfish**
Created by	Dr. Joan Daemen and Dr. Vincent Rijmen	IBM	Dr. Walter Tuchman	Ron Rivest, Adi Shamir, and Leonard Adleman	Dr. Bruce Schneier
Published year	2000	1977	1998	1978	1993
Structure / Scheme	Substitution-Permutation	Fiestel	Feistel	Factoring prime numbers	Feistel
Key length	128, 192, or 256 bits	56 bits	112 or 168 bits	>1024 bits	32–448 bits
Rounds	10, 12, or 14	16	48 DES-equivalent	1	16
Block size	128 bits	64 bits	64 bits	Variable	64 bits
Cipher Type	Symmetric	Symmetric	Symmetric	Asymmetric	Symmetric
Key used	Same key	Same key	Same key	Different key	Same key

TABLE III. Encryption Time Taken for Each Algorithm in Minutes

Files size (MB)	DES ENCR	3DES ENCR	Blowfish ENCR	Hybrid ENCR	Proposed ENCR
64	0.9	0.95	0.92	0.09	0.01
128	1.9	1.95	1.92	0.6	0.1
256	2.9	2.7	2.5	1.6	0.5
512	7.5	6.8	6.1	3.5	1.3
1024	13	12.5	11.8	6.2	5

TABLE IV. Decryption Time Taken for Each Algorithm in Minutes

Files size (MB)	DES DECR	3DES DECR	Blowfish DECR	Hybrid DECR	Proposed DECR
64	1.4	1.2	0.7	0.07	0.04
128	2.3	3.2	1.7	0.3	0.1
256	3	3.2	2.9	1.5	0.4
512	8.4	7.1	5.1	2.8	1
1024	16.4	15.4	12.6	5.7	4.5

The time required to translate ciphertext into plain text is used to compute the decryption time (Table IV). According to the investigation, the suggested model's decryption time is less than that of existing encryption techniques. To decode 1 GB of data, the decryption process takes around 4.5 minutes. DES decrypts a file of the same size in 16.4 minutes, 3DES in 15.4 minutes, Blowfish in 12.6, and hybrid in 5.7 minutes.

TABLE V. Total Time for Each Algorithm in Minutes

Files size (MB)	DES	3DES	Blowfish	Hybrid	Proposed
64	2.3	2.15	2.84	0.16	0.05
128	4.2	5.15	3.62	0.9	0.2
256	5.9	5.9	5.4	3.1	0.9
512	15.9	13.9	11.2	6.3	2.3
1024	29.2	27.9	24.4	11.9	9.5

The Total time is obtained by calculating the time taken to generate a ciphertext from plain text and the time taken to convert ciphertext into plain text . It is observed in Table IV and Fig. 5 that the time taken for the proposed model is minimum for all the files. The total time increases gradually from small to large file size, and for a maximum file size of 1 GB of data, The total time of the proposed hybrid encryption algorithm is 9.5 min, which is 2.4 min less than the hybrid

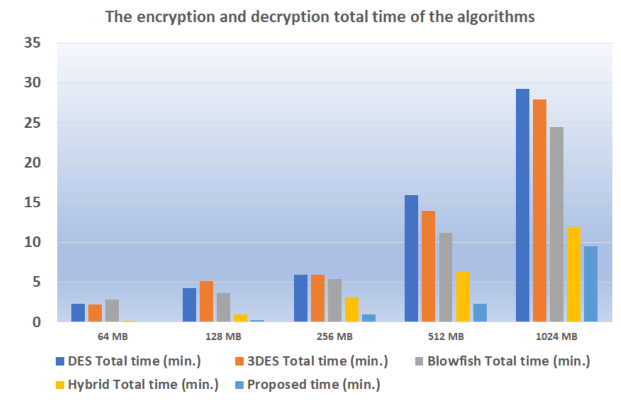

Fig. 5. The encryption and decryption total time of the five algorithms.

algorithm (OTP and AES), 14.9 min less than the Blowfish algorithm, 18.4 min less than the 3DES algorithm, and 19.7 min less than the DES algorithm.

B. Throughput

Throughput is characterized as the quantity of data travelling through a network system. It is the result of dividing all the data delivered in megabytes by the average time required to transfer all the data in minutes. Throughput value in (MB / min) for each algorithm as indicated in Table VI. The throughput study is given in Fig. 6.

TABLE VI. Throughput Value of the Algorithms

Algorithm	DES	3DES	Blowfish	Hybrid	Proposed
Throughput value	34.50	36.07	41.80	88.72	153.20

According to the tests done in this article, the hybrid encryption algorithm may be utilized in software applications, system design and other fields essential for data security exchange, which can effectively safeguard data, in addition to quick performance and execution time, as the results showed that the our approach encryption is 77.48% faster than the DES algorithm , 76.84% faster than 3DES algorithm ,72.71% faster than Blowfish and 42.08% faster than hybrid algorithm.

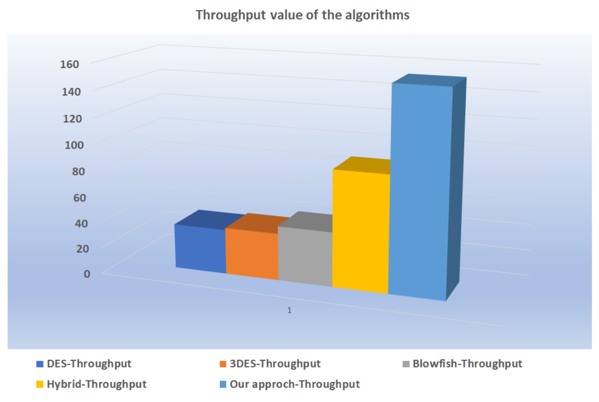

Fig. 6. Throughput value of the algorithms.

VI. Conclusion

In this work, a hybrid encryption method for the Hadoop distributed file system environment's megadata security was presented. When publishing files to the HDFS data node, three-level encryption using the CP-ABE, AES, and RSA algorithms was made possible. The properties supplied for the input file were encrypted using CP-ABE. Additionally, a Rsa key was used to encrypt the key generated using the standard Advanced encryption method.By using this AES key as the password for the encrypted file, data security is increased, and the intruder may be located throughout the decryption process. The effectiveness of the proposed approach has been confirmed in terms of throughput, decryption time, and encryption time. The proposed hybrid encryption model outperformed DES, 3DES, Blowfish, and other conventional Hybrid techniques. The tiny limitation noted in the proposed research is that the CP-ABE approach relies on policies and characteristics, and if the attributes are not selected properly, there may be discrepancies in performance on different runs. Additionally, using optimization techniques to raise other performance metrics may improve the research effort. However, there are promising avenues for future research in this field. Integrating advanced machine learning techniques, exploring post-quantum cryptography, evaluating the impact of blockchain, and addressing scalability challenges are key directions for further investigation. By pursuing these future endeavors, we can strengthen the security of data stored in Hadoop clusters, mitigating emerging threats and ensuring the confidentiality and integrity of sensitive information.

References

[1] Berros, N., El Mendili, F., Filaly, Y. and El Bouzekri El Idrissi, Y., 2023. Enhancing digital health services with big data analytics. Big data and cognitive computing, 7(2), p.64.

[2] 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.

[3] 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.

[4] Hilbert, M., 2013. Big data for development. Retrieved June, 12, p.2019.

[5] 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.

[6] A. Murthy, "APACHE."

[7] Alam, M.B., Hasan, M. and Uddin, M.K., A New HDFS Structure Model to Evaluate the Performance of Word Count Application on Different File Size. International Journal of Computer Applications, 975, p.8887.

[8] Dean, J. and Ghemawat, S., 2004. MapReduce: Simplified data processing on large clusters.

[9] Eman, F.A.O., 2015. S. Abead, Mohamed H. Khafagy,"A Comparative Study of HDFS ReplicationApproaches". the International Journal of IT andEngineering, 3(8).

[10] S. H. Gm, "What is Amazon Web Services." .

[11] 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

[12] 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.

[13] Sean-Philip Oriyano, J. M. Tanna, M. P. Sanghani, M. Ayushi, and R. J. Anderson, "A Symmetric Key Cryptographic Algorithm," Int. J. Comput. Appl., vol. 1, no. 15, pp. 73–114, 2010.

[14] Elminaam, D.S.A., Kader, H.M.A. and Hadhoud, M.M., 2008. Performance evaluation of symmetric encryption algorithms. IJCSNS International Journal of Computer Science and Network Security, 8(12), pp.280-286.

[15] Park, S. and Lee, Y., 2013. Secure hadoop with encrypted HDFS. In Grid and Pervasive Computing: 8th International Conference, GPC 2013 and Colocated Workshops, Seoul, Korea, May 9-11, 2013. Proceedings 8 (pp. 134-141). Springer Berlin Heidelberg.

[16] Lin, H.Y., Shen, S.T., Tzeng, W.G. and Lin, B.S.P., 2012, March. Toward data confidentiality via integrating hybrid encryption schemes and Hadoop distributed file system. In 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (pp. 740-747). IEEE.

[17] Shetty, M.M. and Manjaiah, D.H., 2016, October. Data security in Hadoop distributed file system. In 2016 International Conference on Emerging Technological Trends (ICETT) (pp. 1-5). IEEE.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Abdalwahid, S.M.J., Ibrahim, B.F., Ismael, S.H. and Kareem, S.W., 2022. A New Efficient Method for Information Security in Hadoop. QALAAI ZANIST JOURNAL, 7(2), pp.1115-1138.

[33] Motupalli, R.K., 2022. A Novel Inconsequential Encryption Algorithm for Big Data in Cloud Computing. Journal of Computer Sciences Institute, 23, pp.140-144.