International Journal of Basic and Applied Sciences, 14 (5) (2025) 436-454

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/m46fn971
Research paper

Hybrid Encryption for Fortifying HDFS Data

Shivani Awasthi [1] *, Narendra Kohli [2]

[1] Research Scholar, Harcourt Butler Technical University, Kanpur, UP, India
[2] Professor, Harcourt Butler Technical University, Kanpur, UP, India
**Corresponding author E-mail: awasthisb@gmail.com*

Received: July 11, 2025, Accepted: August 19, 2025, Published: September 14, 2025

Abstract

In the big data era, standard encryption methods alone are not suitable for handling massive, high-velocity data, which negatively impacts the performance of a distributed framework. This paper proposes a hybrid encryption (HE) method that integrates the strengths of the two symmetric algorithms (Twofish-256, AES-256) with the Hadoop Map-Reduce framework (MRF) to fortify Hadoop Distributed File System (HDFS) data. This paper offers dual-level encryption (Twofish -> AES) to mitigate the vulnerabilities of standalone encryption while maintaining optimal performance. The experiments on datasets from 32-256 MB show encryption speed improvement of over 5-6%, efficiency gain of over 5%, and throughput of over 6% compared to hybrid approaches such as CP-ABE+AES, AES+RSA, and standalone encryption schemes AES and Twofish. Additionally, the ANOVA test based on encryption and decryption time gives ($F = 2.67$, $p = 0.07$) and ($F = 9.9$, $p = 0.0003$) outcomes, which show that the proposed HE approach is highly significant in big data environments. Our novel approach balances security and performance, addresses the weaknesses of individual and hybrid encryption algorithms, ensures compatibility in distributed environments, and complies with data protection regulations. This suggested HE approach (Twofish -> AES) complies with GDPR, HIPAA, and PCI-DSS through key management and resistance to side-channel attacks. The results show feasibility in the government and healthcare sectors, where data protection and large dataset processing are critical.

Keywords: *Hadoop; AES; Twofish; Map-Reduce; HDFS; KMS.*

1. Introduction

Enormous data collection is achievable using technologies such as IoT and cloud computing. However, extra storage space and processing frameworks are required for data to work at their best capacity [1-5]. Organizations primarily use MRF and HDFS to process and store large amounts of data in the Hadoop framework. Thus, data security and confidentiality are essential considerations.
A Hadoop system is a solution for processing petabytes of data over millions of nodes in a distributed manner [5]. One of the most fundamental concerns of the Hadoop framework is that it is not always feasible to ensure data protection because data traverses multiple data nodes, which are highly vulnerable to unauthorized access [6-7]. In the second MRF-based concern, an untrusted mapper does not guarantee 100% data security in storage and transit. Therefore, numerous encryption techniques have been used to secure data in the HDFS. However, each encryption scheme has certain disadvantages and is not 100 % capable of securing data in the current era. In a distributed environment (DE) such as HDFS, ensuring data security while maintaining high performance is critical because of the sensitivity and scalability of the data. In DE, multiple nodes exchange data, making them susceptible to unauthorized access during storage and transmission. During data transmission between nodes, risks such as man-in-the-middle attacks, unauthorized access, and cryptographic liabilities increase. An individual algorithm exposes system-specific weaknesses that are inherent to the algorithm. Although AES is efficient, it can be susceptible to side-channel attacks if it is improperly implemented. Although Twofish is highly secure, it may not deliver the speed required for large-scale data encryption applications. In DE, large-scale data encryption and decryption often lead to high latency and block HDFS capabilities. Therefore, it is necessary to balance encryption strength and computational efficiency. The proposed HE combines Twofish and AES to address the security and performance challenges in DE. In terms of security, layered encryption increases the overall flexibility of the system because compromising both encryption layers concurrently is significantly more difficult. In terms of performance, AES, known for its hardware acceleration abilities, offsets the computational overhead presented by Twofish and guarantees efficient encryption processes.
In this way, this proposed approach fulfills the research gap that is as follows: “inability of current encryption techniques and cascading encryption approach to provide security and efficiency.”

1.1. Limitation of Cascading Encryption

The current cascading encryption approach suffers from three shortcomings in the Hadoop framework
1) Sequential Processing Framework

Copyright © Shivani Awasthi, Narendra Kohli. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The sequential process followed by the cascading approach gives 20-35% throughput degradation with respect to standalone encryption (Filaly et al., 2023); however, our hybrid approach provides quadratic results based on the data size [8].

2) Key Management Overhead

Each algorithm key is distributed individually, increasing the synchronization complexity $O(n^2)$ of n nodes (Sunder et al., 2022) [9].

3) Intermediate Data Exposure

In the encryption stage, data exists in decrypted form, which is prone to high risk and creates vulnerabilities during the MRF.

In Hadoop, each node has storage and processing power, and its structure comprises clusters of computers. Large data are partitioned into data blocks processed by MapReduce, and after processing, the data are saved in HDFS in less time (White, 2015; Borthakur, 2007). The open nature of the Hadoop framework requires protection at the storage and processing (mapper) level. This is the main motive for developing the encryption approach to protect the data and also to be compatible with the Hadoop system for providing high performance in handling large datasets [1], [6].

1.2. Different Proposed HE Solution

1) Parallelized Dual Level Encryption

Each block is processed with both encryption methods within each mapper in a big-data environment.

2) Unified Security Architecture

The hybrid encryption approach, which follows the unified security architecture, has the features listed in Table 1. Table 1 highlights how our proposed HE approach provides better outcomes than other encryption approaches.

Table 1: Unified Security Architecture Feature

Features	Method	Our Hybrid Encryption solution
Key distribution	Separate channel used	Single PLE/TLS used
Attack resistant	Brute force attack	Side Channel + Brute Force attack
Throughput	1124MB/Sec	1191 MB/Sec

3) HDFS Optimization Implementation

The encryption concept is implemented in the map task using a hybrid approach. Thus, the shuffle phase I/O is reduced by 35 % through the compressed concept.

Thus, the suggested hybrid encryption solution (Twofish, AES) provides better encryption speed and throughput by using the hybrid encryption solution advantages of Twofish and AES with MRF to secure data compared with other conventional hybrid solutions and standalone encryption methods. The proposed HE approach also complies with the GDPR/HIPAA requirements for sensitive data processing.

Providing robust encryption is critical in the government, healthcare, and finance sectors. In the healthcare sector, HIPAA-compliant protection for patient records and the financial sector, GDPR, are used to protect sensitive data. Current standalone encryption fails to balance security and performance requirements, making it vulnerable. Our proposed HE approach addresses these gaps by integrating the resilience of Twofish with the efficiency of AES for real-world applications.

The novelty of the proposed HE, Twofish with AES, is as follows:

1) Unlike standard encryption, which often fails to meet the large amount of data processing requirements with untrusted MRF in the Big Data field, our approach leverages the efficiency of parallel data processing with MRF. HE with MRF ensures that the overall process does not delay large data encryption and decryption or cause any scaling issues.
2) The layered approach enhances data security through data encryption using Twofish and AES. If the attacker compromises one layer, they must also bypass the other layer, which is highly impossible because of the cryptographic design. Twofish protects against side-channel attacks, whereas AES protects against brute-force and cryptographic attacks. Thus, a layered approach enhances security. Therefore, there are two layers in the proposed HE approach for securing data, thus making it even more immune to any possible attacks that may act against it for any reason.
3) The evaluation results demonstrate that the proposed HE approach improves the encoding speed by more than 5-6% and throughput by more than 6% compared with conventional encryption and other hybrid encryption methods.

This paper is divided into the following sections: literature review, methodology, results and discussion, and conclusion.

2. Literature Review

Previous research has focused on analyzing cascade and standalone encryption methods to enhance the security of cloud data. These studies followed multiple encryption strategies to enhance data security in cloud environments. Filaly et al. (2023) recommended hybrid AES, CP-ABE, and RSA-based information security algorithms for securing Hadoop data. However, the question of how to resolve the scalability problem of this approach to encrypt and process large amounts of data remains open in this study, as it has not been addressed. Therefore, this study limited the applicability of the approach in a big data environment [8]. Awasthi et al. (2022) particularly focused on fortifying Big Data in Hadoop using hybrid encryption (RSA, AES) and key management concepts. This study addresses the limitations of AES encryption but does not address the performance trade-off in practical deployment [9]. Similarly, Viswanath and Krishna (2021) focused on hybrid encryption in a multi-cloud framework to protect large amounts of data from unauthorized access. The authors used a private cloud to securely store the data; however, cost factors and scalability issues remain to be addressed [10]. Negi et al. (2023) developed AES and ECC a combined cryptographic approach, to protect files in the cloud storage. The main limitation of this study was the cost [11]. Similarly, Kumari and Malhotra (2020) and Lai et al. (2022) studied cloud storage using a hybrid cryptography concept; however, they did not implement the proposed approach efficiently in a real-time environment and still have open performance-related issues [12-13]. This study provides only a theoretical analysis framework. Chaundry et al. (2023) presented a study related to hybrid cryptography for secure files, emphasizing their significance and effectiveness [14]. However, this study lacked practical deployment and performance metrics. Furthermore, Mohanraj et al. (2022) used CP-ABE and AES to preserve data in HDFS. This study addresses access control and encryption issues. The drawback of this study is that the specific qualities of the rules and methods have been previously defined [30]. Kadre et al. (2015) proposed an AES-MR method to secure data in HDFS; however, this paper uses AES alone, so the possibility of vulnerabilities of numerous attacks on stored and transit data is maximum. However, this approach lacks a layered security structure [31]. Du et al. (2019) highlight the performance of the Hadoop framework when dealing with the map-reduce processing. In this study, many intermediate data are generated in the map-reduce task, and a large amount of time is required for disk I/O processing and network transmission time. In this

study, the author used a compression method with hardware acceleration to save time for disk I/O and network transmission overhead. This study is not feasible or practical in a real-time environment because of its hardware dependency [37]. On the other hand, Jain et al. (2019) and Gupta et al. (2023) proposed a Secure Map Reduce layer and a Fortified Secure Map Reduce framework to provide a safeguard data pipeline, but both approaches degrade overall performance based on their structural complexity [15-16]. Mahmoud et al. (2018) proposed AES with OTP for cloud data security. This study had transmission and storage overhead problems owing to the large key size required for the OTP [32]. However, this approach enhances data security in big data environments. Zhang et al. (2020) used a blockchain method to enhance security; however, their study had high infrastructure complexity and poor scalability [40]. (Mothukuri et al., 2021) provided the BlockHDFS framework to provide security and traceability features to secure data in HDFS, but only the top-level directory is encrypted and decrypted [39]. This study integrates encryption and authentication technologies for an intelligent software protection system in a Spark environment. By doing this, the system greatly enhances the performance and accuracy when facing an attack by 2.14% [41]. The drawback of this approach is the complexity and implementation overhead for designing the above model. A hash function leveraging Spark and a two-dimensional coupled dynamic integer tent map was advanced to overcome security and efficiency challenges associated with processing large volumes of data. Initially, the plaintext is loaded and divided into the Spark platform, followed by parallel processing. A Merkle tree structure is implemented to consistently compress fixed-length data, utilizing a two-dimensional coupled image lattice with a dynamic integer tent map, along with additional dynamic parameters to boost performance. The parallel processing capability of Spark significantly enhances the operational efficiency of the algorithm [42]. The drawback of this approach is its high structural complexity. An overview of the related work and various aspects of HE in cloud environments is provided in Table 2.

Table 2: Study of Related Work

Study/Method	Encryption Techniques	Security Level	Performance	Scalability	Comments	Comparative insights
Filaly et al. (2023)	AES, CP-ABE, and RSA	High: Number of encryption layers	Moderate: huge overhead due to many algorithms.	Limited: Less Scalable.	Lack of Scalability and efficiency.	Fail to optimize performance Not scalable in the MapReduce framework
Aswathi et al. (2022)	RSA, AES	High: Double security layer-based high security through RSA and AES.	Moderate: A large RSA key affects the Performance	Limited: scalability issue	This cost-effective approach provides robust security and is less scalable.	Fail to optimize performance Not scalable in the MapReduce framework
Viswanath and Krishna (2021)	Hybrid Multi-Cloud, AES	High: Security.	Moderate: depends on cloud performance.	Limited: private cloud used.	Scalability issue.	Lack of practical feasibility Not scalable in the MapReduce framework
Negi et al. (2023)	ECC and AES	High: ECC-based Strong encryption support.	Moderate: Computational complexity is high owing to ECC encryption.	Moderate: limited scalability.	Large-scale data processing still faces issues.	Fail to optimize performance Partial scalable
Lai et al. (2022); Kumari and Malhotra (2022)	(AES+ others) Hybrid cryptography	High: several algorithms based on high-security	Moderate: specific hybrid approach performance differs.	Limited: Not implemented in real-world scenarios.	This approach is not scalable to the real world.	Lack of practical feasibility Not scalable
Vadre et al. (2015)	AES-MR	Low: Side-channel attack, brute force attack chance is maximum.	Good: Standard encryption provides better results.	Moderate: scalable	Unauthorized access is maximized due to known attacks and a small key size	Poor security Moderate performance and scalability
Jain et al. (2019)	Secure Map-Reduce	High: extra security layer for MapReduce	Moderate: increase extra overhead due to additional layers	Limited: scalability and complexity issues	Adding an extra layer of security would increase complexity and scalability issues.	Not scalable Fail to optimize performance
Gupta et al. (2023)	Fortified Secure Map-Reduce	High: Improved security over SMR	Moderate: performance issues	Limited: Complexity cuts scalability	Scalability limitations.	Not scalable Lack of practical feasibility
Du et al. (2019)	Hardware compression accelerator	Moderate: security	High: performance	Limited: scalable	Scalability limitation	Not scalable Lack of practical feasibility
Mahmoud et al. (2018)	AES, OTP	High: security level	Moderate: transmission and storage overhead	Moderate: scalable	The key size is as long as the data, so considerable transmission and storage overhead occur.	Performance bottleneck Moderate scalable
(Zhang et al., 2020)	Blockchain + HDFS	High-security	Slow: performance is slow	Limited: less scalable	High infrastructure complexity	Optimized performance
(Mothukuri et al., 2021)	Blockchain +HDFS	High-security	Slow-performance	Limited-less scalable	Only the top-level directory is encrypted and decrypted	Optimized performance
Xu and Li (2024)	Encryption + Authentication scheme	High security	Moderate performance	Moderate- scalable	Computational overhead and latency are high	Optimize performance

(Liu et al. 2023)	Hash function with a dynamic integer	High security	Slow performance	Less scalable	Cost and speed are limitations	Less scalable
Proposed Hybrid Approach (Twofish + AES)	Twofish, AES	High: Double encryption increases the resilience	High: Optimized Map-Reduce performance	High: Leverages Map-Reduce for scalable processing.	Balance performance, security, and scalability to make it suitable for big-data environments.	Optimized performance Scalable in map-reduce framework Practical feasible

Most existing hybrid encryption models lack real-world deployment, suffer from high computational overhead, or fail to scale efficiently in distributed environments, such as Hadoop. The proposed HE technique not only fills the recognized gap in the above literature but also overcomes AES data security-related issues when used alone in DE and other hybrid encryption challenges. This scheme also boosts the encryption speed and throughput without compromising operational efficiency when using an MRF.

2.1. Performance Comparison of Prior Hybrid Encryption Approaches and Proposed HE Approach

Table 3: Prior Hybrid Encryption Approach and Proposed HE Approach Comparison

Study/Method	Encryption Techniques	Throughput	Latency	Speed Improvement
Filaly et al. (2023)	AES, CP-ABE, and RSA	845	220	-25%
Aswathi et al. (2022)	RSA, AES	910	195	-15%
Viswanath and Krishna (2021)	Hybrid Multi-Cloud, AES	980	180	-8%
Negi et al. (2023)	ECC and AES	875	210	-20%
Lai et al. (2022); Kumari and Malhotra (2022)	(AES+ others) Hybrid cryptography	1020	165	-5%
Vadre et al. (2015)	AES-MR	1124	140	Baseline
Jain et al. (2019)	Secure Map-Reduce	935	2001	-17%
Mahmoud et al. (2018)	AES, OTP	890	215	-21%
Du et al. (2019)	Hardware Compression Accelerator	978	234	-13%
Zhang et al. (2020)	Blockchain + HDFS	887	245	-15%
(Mothukuri et al., 2021)	Blockchain + HDFS	898	278	-12%
Xu and Li (2024)	Encryption + authentication scheme	895	267	-7%
(Liu et al., 2023)	Hash +choes	879	277	-9%
proposed HE	Twofish, AES	1278	256	+6% or more

The outcomes in Table 3 highlight that our suggested HE approach with MRF is used to protect the data and boost the performance when dealing with large datasets with Twofish resilience and AES encryption speeds.

3. Methodology

The suggested HE, Twofish with AES using MRF [26], enhances HDFS data security and optimizes the encoding and decoding speed of HDFS stored data. This dual-layered protection mitigates individual limitations.

3.1. Justification for The Hybrid Method

The choice to combine Twofish and AES within MRF is justified based on the exact requirements of secure data in the HDFS

3.1.1. Performance and Security Balance

The Twofish hybrid approach with AES balances the need for high performance and robust security features. Twofish is secure against side-channel attacks and has a high computational cost. AES is fast for data encryption and decryption, but it may not offer better security in certain cases. Using both, the strategy ensures that the system benefits from the security of Twofish and the performance of AES, thereby providing an excellent solution for HDFS encryption. The use of symmetric (Twofish, AES) algorithms within the MRF framework allows the encryption process to be distributed across multiple nodes, thereby improving the overall performance.

3.1.2. Approaching Individual Algorithm Weaknesses

Encryption algorithms have limitations. Twofish is highly secure but slower and more difficult. AES can be weak against specific attacks if used alone. In the current era, the hybrid method mitigates these problems by assembling encryption, thus minimizing the possibility of an attacker exploiting any situation.

3.1.3. System Compatibility

Both the Twofish and AES encryption algorithms are well-known and suitable for MRF-based Hadoop systems. This agrees with earlier acceptance without requiring changes to the existing structure.

3.1.4. Scalability on A Distributed Platform

The MRF is built to process vast amounts of data across distributed nodes. With a hybrid approach within the MRF, the process can easily scale with immense data, guaranteeing that encryption does not slow down the system's growth.

3.1.5. Regulatory Compliance

The proposed method aligns with data protection standards, such as GDPR, HIPAA, and PCI-SS, by demonstrating HE, which lowers the probability of data breaches (Table 4).

Table 4: Adherence of Data Protection Laws

Regulation	Relevant Requirement	How the Hybrid Encryption Approach Adheres	Impact in the real world
GDPR	Data Protection by Design and by Default (Article 25)	Ensuring data protection throughout its lifecycle by integrating Twofish and AES.	All stages of sensitive data are secured, thereby reducing the risk. In 2023, the German health care system will require that patient records be encrypted through double-layer encryption. If one layer is compromised through AES, a side channel attack, the second layer protects the data. This feature demonstrates the layered encryption value.
	Security of Processing (Article 32)	Implementing strong encoding (Twofish+ AES) for HDFS and processing.	Robust encryption is provided to defend against illegal access and potential data breaches. The finance institute used this suggested HE approach to secure transaction logs. The bank reduces the encryption latency while maintaining state-of-the-art security through GDPR.
	Data Breach Notification (Articles 33 and 34)	Encryption guarantees that the data is unreadable, possibly freeing it from notification requirements.	Decrease regulatory burden by justifying the need for breach notifications when data is accessed. Dutch e-commerce company (2022) avoided a fine after a breach because the attacker was not able to decrypt the (Twofish+AES) hybrid encryption-based customer data.
	Data Access and Portability (Articles 15 and 20)	Encrypted data controls access and guarantees secure data portability during transfer.	It protects sensitive data during access and transfer, aligning with the portability and access rights of GDPR. The European health portal provides patient access to the data. Data encrypted with suggested HE, so this HE enables on-demand role-based decryption. The finance sector Fin tech customer takes the data about the transaction history. The bank sector used HE (Twofish+AES). This ensures the secure portability to meet GDPR compliance.
HIPAA	Security Rule (45 CFR Part 164 Subpart C)	Twofish and AES are used for encryption to maintain electronically protected health information (ePHI).	This meets HIPAA's requirements for technical safeguards and confirms ePHI security. A U.S.-based telemedicine company encrypts patient video and diagnostic information through HE. Twofish is a defense against GPU-based side channel attacks with AES real-time encryption, so HE is able to provide the video 4K subscriber without any interruption. Doing this, HE (AES+Twofish) follows the security rules of HIPAA without violation.
	Breach Notification Rule (45 CFR §§ 164.400-414)	Encrypt the standards of ePHI to HIPAA, possibly qualifying for the safe harbor provision.	Decrease breach notification obligations if encrypted data is cracked. During an incident in 2021, a HIPAA-covered organization managed to evade penalties for breach reporting since the attackers accessed only AES-encrypted data, while the backups encrypted with Twofish remained secure. This confirmed the importance of the hybrid approach's "safe harbor" facility.
	Data Integrity (45 CFR § 164.312(c))	Ensuring ePHI integrity by preventing unauthorized changes through encryption.	Maintains the reliability and accuracy of ePHI, fulfilling HIPAA's integrity requirements. A clinical research center used HE to encrypt trial data. Twofish and AES authentication tag, Twofish and AES-GSM ensured tamperproof logs meeting the data integrity rule of HIPAA.
	Key Management and Access Control (45 CFR § 164.312(a))	It implements secure key management, including access control and key rotation.	Protects encryption keys, ensuring that only users have access to ePHI, in line with HIPAA's safeguards. A US hospital chain used Twofish and AES. Twofish-256 key store in HSM-Backstore KMS and AES-256 stores in Hadoop KMS. 2023 ransomware attack-based AES key compromised, but Twofish key remains secure.
PCI-DSS	Encryption of Cardholder Data	Strong encryption standards (Twofish + AES) are used for protecting cardholder data.	Guarantees cardholder data is encrypted securely, meeting the requirements of PCI-DSS encryption. A payment processor company used HE to encrypt cardholder data 10M/month. Twofish key flexibility and AES speed feature give encrypted data with minimum encryption overhead and follow the PCI-DSS standard.
	Secure Transmission of Cardholder Data	Robust encryption during network transfer.	Protects cardholder data during transmission, which conforms to PCI-DSS requirements for data security. In real-time payment gateway processing, this HE ensures protection if the transmission channel is compromised, the cardholder's data remains unreadable without the attacker getting keys.

In this study, the integration of Twofish and AES guarantees that encrypted data satisfies the rigorous requirements for protecting sensitive information, particularly in the healthcare and financial sectors. This novel approach protects sensitive data throughout its lifecycle, reduces the impact of potential data breaches, and satisfies the requirements of various regulatory frameworks. The innovation of the suggested strong HE solution lies in the strategic mixing of Twofish and AES within the MRF, which is specifically tailored to address the issues of safeguarding large-scale data in HDFS. Thus, this novel technique not only boosts data security but also improves speed, scalability, and compliance with data protection rules, making it an important contribution to the field of big data security in the modern era.

3.2. Reason for Algorithm Selection

In this study, we combined (Twofish and AES) algorithms based on different criteria, as shown in Table 5.

Table 5: Twofish and AES Algorithm Selection Criteria

Criteria	Twofish	AES	Justification
Security	High: High resistance to side-channel attacks	Moderate: High resistance to brute-force attack	Twofish provides robust defense against side-channel attacks, and AES ensures moderate security; thus, strong overall security is achieved.
Performance	Moderate: Slow due to complex key schedule	High: Fast, especially with hardware support	AES provides high-speed encryption, which makes it suitable for large-scale data operations.
Compatibility with HDFS	Good: Compatible with symmetric encryption	Excellent: Widely adopted and supported	The proposed algorithms are compatible with HDFS
Resistance to Specific Attacks	High: High against side-channel attacks	Moderate: Requires careful operation to resist side-channel attacks	The Twofish resistance to side-channel attacks balances the strength of AES with other cryptographic methods.
Flexibility and Adaptability	High: Flexible key schedule and structure	High: Broadly supported and optimized	Both algorithms offer flexibility, with AES providing broader hardware and software support.
Overall, Balance	High security with moderate speed	High speed with moderate security	This combination balances the need for both high security and high performance, making it ideal for HDFS.

In summary, the integration of Twofish and AES in the proposed HE approach provides a balanced solution that satisfies the security and performance requirements of HDFS. Twofish enhances the system's defense against specific attack vectors, such as side-channel attacks, but is slower, whereas AES ensures that the encryption process remains efficient and compatible with the existing infrastructure. This careful selection ensures that the system can handle difficulties associated with massive data processing without compromising security and meet NIST SP 800-175B rules for providing layered security.

3.3. Twofish and AES Integration

This hybrid encryption solution secures the massive amount of data stored in the HDFS using two symmetric algorithms (Twofish and AES). Compared to single-level encryption technologies, double-level encryption procedures provide advanced data safety. Twofish delivers robust encryption abilities, whereas AES maintains CC (compatibility and competence) [17-21]. Twofish provides better results based on the increasing RAM size in the Hadoop framework [34], and AES already performs better; therefore, this integration gives better results as a proposed HE with MRF. The cryptanalysis summaries of both Twofish and AES are listed in Table 6.

Table 6: Cryptoanalysis Summary of Both Twofish and AES

Cryptographic Aspect	Twofish	AES	Comparison
Algorithm Type	Feistel network-based symmetric block cipher	Substitution-permutation network-based symmetric block cipher	Here, both are symmetric block ciphers.
Block Size	128 bits	128 bits	Equal block sizes ensure compatibility in hybrid schemes.
Key Sizes	128, 192, and 256 bits	128, 192, and 256 bits	Similar key size flexibility
Number of Rounds	16 rounds	10, 12, and 14 rounds (depending on key size)	Twofish uses a fixed 16 rounds while AES uses 10/12/14 rounds, making Twofish potentially more resistant to certain attacks.
Security Against Brute-Force Attacks	Strong: 2^{128} for 128-bit keys	Good: 2^{128} for 128-bit keys	Both offer robust protection against brute-force attacks, but Twofish has strong resistance.
Resistance to Known Attacks	High: No known practical attacks	High: No known practical attacks	Both are highly resistant to known attacks; however, Twofish offers additional structural security.
Resistance against Side-Channel Attack	High: designed to handle side-channel attacks	Good: Weak timing and power analysis attacks	Twofish is more resistant to side-channel attacks, whereas AES requires careful implementation to enhance its performance and security.
Performance (Speed)	Slower due to the complex key schedule and structure	Faster due to efficient design and hardware support	AES is faster in terms of software and hardware implementations, making it more suitable for high-speed applications.
Memory Requirements	Moderate: Uses precomputed key-dependent S-boxes	Low: Efficient memory usage with fewer precomputation	AES has lower memory requirements, which makes it more suitable for resource-constrained situations.
Flexibility and Adaptability	High: Flexible key schedule and S-box structure	High: Widely supported across platforms and optimized for hardware acceleration	Both are flexible and adaptable; however, AES has broader hardware support and optimization.
Cryptanalysis History	Extensive: Resilient to cryptanalysis attempts since 1998	Extensive: Carefully analyzed since its adoption in 2001	Both algorithms have withstood extensive cryptanalysis, reinforcing their reliability in secure applications.
Use Cases	Best for high-security environments with potential side-channel risks	Best for environments requiring high speed and broad hardware support	Twofish is preferred in environments where side-channel attacks are a concern, whereas AES is preferred in high-performance, hardware-accelerated applications.

A summary of the cryptanalysis is presented in Table 6. Twofish is based on the Feistel network, and AES is based on a permutation and substitution network with high-performance and high-speed features. Twofish resists cryptanalysis through 16 rounds, whereas AES performs in 12-14 rounds. Both algorithms are resilient to brute-force attacks and other known attacks. Twofish, owing to its complex key schedule and key-dependent S-box, is resistant to side-channel attacks. In addition, AES requires careful implementation to resist timing and power analysis attacks. This makes Twofish more secure, where the physical risk is higher. From a performance perspective, AES is faster because it requires less memory in a distributed environment, whereas Twofish is slower but improves the cryptographic strength. Finally, the proposed HE overcame each other's limitations and provided a balanced solution that enhanced security without compromising

operational efficiency through layer integration. The complementary nature discussed in Table 6 validates the discussion that this suggested HE approach is suitable for a distributed environment

3.4. Algorithm Integration Structure

In this proposed HE, the integration is based on the sequential layering of both encryption algorithms, Twofish and AES, to mitigate the weaknesses of each encryption algorithm. Twofish structures ensure a high level of security, with no successful practical attacks. AES is used for fast encryption and decryption because of its design characteristics. AES is universally supported by processors to ensure minimal overhead. AES's simplicity of AES makes it ideal for high throughput. In each mapper, Layer 1 encrypts data using Twofish-EAX to prevent side-channel attacks, feeds the output to AES-GSM encryption for optimal hardware acceleration, and produces a ciphertext with a dual authentication tag. In the proposed HE, both encryption algorithms use a 256-bit key via SCRYPT KDF. The NONCE is handled efficiently by the RANDOM BYTE generator. The authentication ciphertext is a combination of the cipher tag with both the NONCE (Twofish, AES) and the tag (Twofish, AES). In the suggested HE approach, data are encrypted through each mapper to provide security of HDFS data locally, and then shuffling is performed via the reducer, and the key management server (KMS) provides automatic key rotation; thus, the suggested HE approach improves security in DE. The test results show a throughput increase of more than 6% compared to the traditional and cascade encryption methods. This HE also meets regulatory compliance through a dual-authentication mechanism. This approach differs from the cascade and parallel approaches by creating cryptological interdependence between the two algorithms while maintaining the performance characteristics that are essential for big data environments.

The suggested HE approach has some advantages over standalone encryption

1) Both encryption algorithms use different methods that require attackers to crack both algorithms simultaneously; thus, the suggested HE approach provides better security.
2) Performance is maintained by the AES-GSM mode with hardware acceleration features, which helps with the Twofish encryption approach.
3) MRF-based optimized processing is used because the standalone encryption speed of parallel encryption is limited; thus, based on the suggested HE approach, an optimized performance speed can be achieved.

3.5. Hadoop Distributed Environment

In 2006, Hadoop became part of Lucene's sub-project, which included a distributed system structure. In addition to being a distributed storage system, Hadoop has massive data processing capabilities [22]. Hadoop is a distributed framework deployed on low-cost hardware devices. This framework operates in the HDFS and MRF [23-25]. The MRF is built on a parallel processing concept for enormous amounts of data processing [23]. Both map and reduce are crucial properties of the MRF. The MRF operates on key-value pairs. The MRF performance depends on the adjustment of the memory, CPU allocation, and the number of MapReduce tasks. Thus, the MRF provides better results without compromising its operational efficiency. In Hadoop, the input file is first split into blocks and stored in a data node. The Hadoop architecture works on the name node and data node concepts. All operations were performed between the two nodes. The MapReduce architecture process is as follows: At the start, the client sends an access permission request to the name node. If this request is accepted, the HDFS block ID, where the blocks of the file are stored, is returned. The client receives the block ID list, where the file part is saved [26]. MRF uses a mapper and reducer to process files. The mapper converts the file into key-value sets [27]. This key-value set is forwarded to the new partition and sorts the data constructed on the keys. The combiner integrates the sorted key-value pairs and counts the same key values. Finally, the partitions as a key-value set are again offered to the reducer. Map-Reduce uses a shuffle operation to pass the outcome of the mapper to the exact reducer. The reducer then uploads the results to the HDFS. Fig. 1 represents Ahmed et al. (2020) Hadoop MapReduce architecture [28].

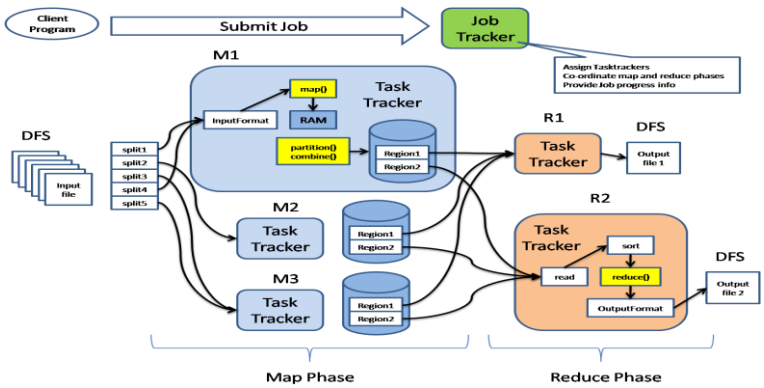

Fig. 1: Hadoop Map-Reduce Architecture Ahmed et al. (2020) [28].

3.6. Justification of MRF Used in The Proposed HE Scheme

The main motive for using the MRF with the suggested HE approach is to achieve high parallelism, scalability, and localized encryption efficiency in the DE. Below is the justification that validates the MRF as an optimal framework for secure big-data encryption with efficient processing.

3.6.1. Distribute Parallel Processing at The Block Level

The MRF is used for parallel processing of data in DE, where a large file is split into independent blocks. Each block is processed by an independent mapper. Each mapper works to secure the data using the suggested HE logic to encrypt the data. Twofish with the AES concept performs in parallel with the MRF to reduce the encryption time. In addition, the centralized server is replaced with a traditional encryption framework capability for the parallel processing feature. In DE, Hadoop YARN is used to reassign the failed task to maintain reliability and uses MRF for secure, scalable data processing across multiple independent nodes.

3.6.2. Reduced I/O Overhead via in-Mapper Computation

In the traditional encryption framework, data are encrypted pre-storage, resulting in redundant disk I/O; however, through MRF, the unencrypted intermediate data written to the disk are reduced through the mapper task. In addition, I/O is reduced in the shuffle phase by partial in-memory compression post-encryption. Thus, the MRF framework minimizes latency and supports real-time data processing in DE.

3.6.3. Integration of Encryption within Existing Data Pipelines

Most companies rely on the Hadoop framework for MRF-compatible ETL and analytics jobs. Integrating the suggested HE in the MRF ensures that processing is performed securely without any change in architecture. This integration is also compatible and synchronized with the existing workflow. MRF is easily compatible with the Hadoop 2 and 3 ecosystems.

3.6.4. Fault Tolerance and Job Tracking

If any node is compromised, the MRF is used to re-encrypt the block without any loss of key via the KMS. Block tracking is performed through a job tracker without any loss of key via KMS. This feature ensures encryption reliability in the DE.

3.6.5. Proven Scalability and Cluster Adaptability

The experiment was conducted on Hadoop 3.x with four executors, each with two cores and 8 GB RAM. Input files ranging from 32 to 256 MB were used for processing in four mapper nodes.

3.6.6. Performance Test Demonstrates

Using MRF in the Hadoop environment to enhance high throughput and minimize latency over other encryption approaches existing in DE.
In summary, the use of MRF improves performance and security with secure, scalable, and fault-tolerant encryption. The MRF's compatibility with HDFS and KMS makes it ideal for securing data using the suggested HE in a big data environment.

3.7. Encryption Method

In the encryption process, the data are first split into manageable chunks or parts, which is essential for processing in the MRF. Each data chunk is first encoded by Twofish, which supports key lengths of up to 256 bits and a 128-bit block size and is recognized for its security and flexibility. Once the data chunks are encrypted with the Twofish algorithm, the resulting ciphertext is encrypted again using the AES algorithm. AES is fast and efficient, thus quickly encrypting vast amounts of data. It has a 128-bit block size and can utilize 256 bits of key size. In dual-layer encryption, Twofish followed by AES guarantees that an attacker faces the challenge of decrypting the data; thus, an overall security improvement is possible. In an MRF, the encryption process can be distributed across multiple nodes. The mapper function is responsible for processing the data blocks, applying the Twofish encryption and then AES, and then passing the encrypted data to the reducer function. The reducer sort combines all data output from the map task and sorts it. This parallel processing reduces the encryption time. This approach is feasible for handling huge datasets and provides robust security. The encryption process is illustrated in Fig. 2.

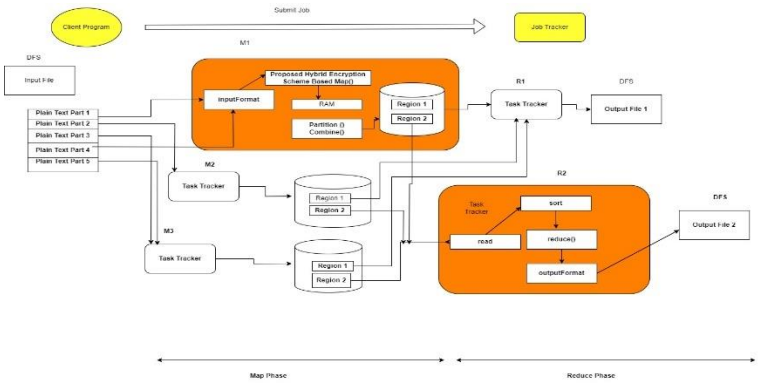

Fig. 2: Proposed Hybrid Encryption Scheme.

3.8. Decryption Method

The data decryption process mirrors the encryption sequence in reverse order. The KMS provides each mapper with an HE-based key, which is the integration of Twofish and AES. The HDFS contains the encrypted form of text (ciphertext) in the Hadoop distributed environment. Each encrypted data is processed in reverse order by the mapper and reducer through the MRF. Each chunk of encrypted data is sent to each mapper, which then decrypts the encrypted data based on the proposed HE approach. Each mapper first decrypts the data using AES and then Twofish based on the key provided by the KMS server. The output in the form of chunks of plaintext data is sent to the reducer, which shuffles, sorts, and combines the chunks of plaintext data and provides the plaintext data. Finally, the output is again sent to the HDFS, where the client again accesses the data. This two-step decryption method ensures that the data remains secure throughout its lifecycle in the HDFS and provides good efficiency; if one layer is compromised in the decryption process, the second layer provides data security. The use of the proposed HE not only offers robust security against data crackers but also allows for the safe transfer of keys via the KMS server. The decryption process is shown in Fig. 3.

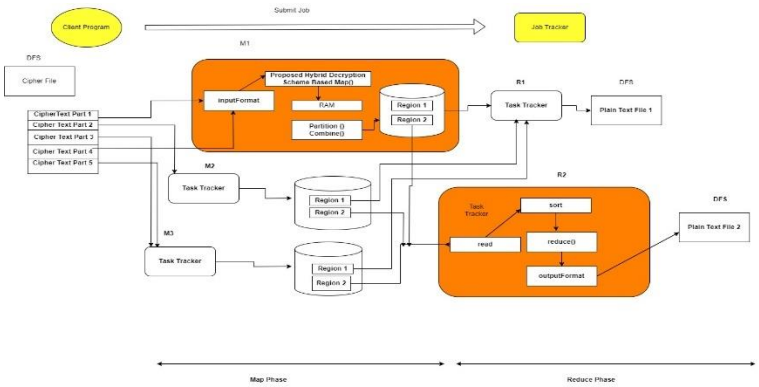

Fig. 3: Proposed Hybrid Decryption Scheme.

3.9. Key Management

The proposed model employs a key management system to generate, distribute, and secure encryption keys for both the Twofish and AES algorithms. The security of the encryption keys is guaranteed using key rotation and revocation mechanisms. Both encryption methods generate keys 256 bit using a random number generator to ensure high entropy and resist brute-force attacks. Keys that include metadata such as creation time, expiration time, and access permission. The key distribution is based on the PKI X. 509 key distribution certificate and TLS frameworks to prevent man-in-the-middle attacks [35-36] and ensure confidentiality, integrity, and authenticity. Only authenticated mapper/reducer node requests take the encryption key. PKI+TLS offers mutual authentication and secure session establishment. This is essential in the Hadoop framework, where nodes are dynamically allocated in multi-tenant and virtualized environments. TLS ensures that if a session is compromised, the past transmission is still secure. TLS is used efficiently in symmetric key exchanges. The symmetric key exchange used TLS once per session and was then cached at the mapper. KMS easily integrates with the Hadoop framework to ensure scalability and enhance performance. Key rotations enhance security and mitigate the risk of key compromises. To reduce key exposure risk, automated time and usage-based rotation policies are necessary. Keys are rotated every fixed number of jobs and after a set time threshold. The KMS reissues the new key and updates the mapper node cache. If the key and node seem to be compromised, the new key is reinvoked, and the re-encryption is triggered using the new IDs. The reinvocation list is stored in the Namenode and propagated to all active data nodes. The Key Management Server (KMS) performs all the above operations; thus, key security in KMS and Hadoop storage is necessary [35]. The keys are distributed in parallel among all the nodes to encrypt the data blocks. To ensure reliability, the KMS is implemented with a redundant master slave architecture with a Zookeeper for failover support and coordination. If a KMS failure occurs, backup assistance takes over without compromising the data encryption processing. Additionally, caching short-lived keys on the mapper side guarantees temporary continuity even during brief interruptions in the encryption process. In this way, through the proposed approach with efficient management of a large volume of keys, data are secured between an untrusted mapper and reducer across the Hadoop framework. Fig. 4 Shows The life cycle of the proposed KMS framework.

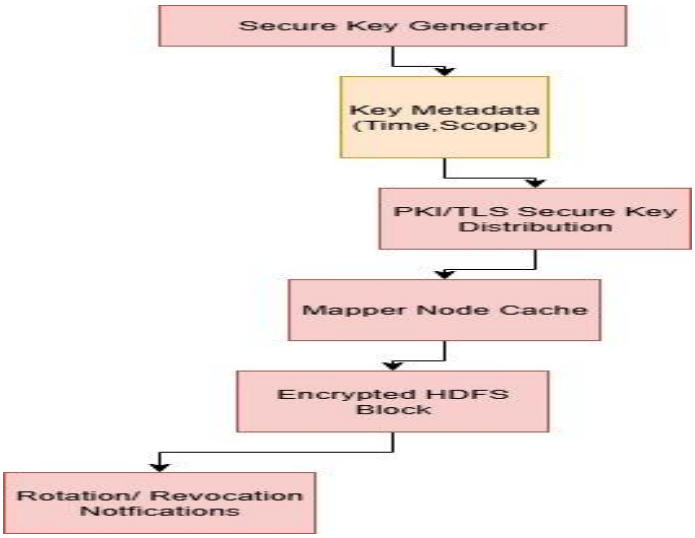

Fig. 4: Key Life Cycle Diagram.

3.10. Security Analysis and Mechanism

In this study, HE (Twofish, AES) was used, and the security analysis based on the different characteristics of Twofish and AES is presented in Table 7.

Table 7: Security Analysis Based on Different Security Characteristics

Security Aspect	Twofish	AES	Experiment Analysis	Treat Model
Cryptographic Strength	High: brute-force and cryptanalysis resistant	High: brute-force and cryptanalysis resistant	The two algorithms were confirmed to be effective against cryptographic attacks (brute-force, differential cryptanalysis).	Threats: attack based on brute force, differential cryptanalysis. Mitigation: They are resistant to both types of attacks
Resistance against Side-Channel Attack	High: complex structure-based high resistance	Moderate: Careful implementation needed	Twofish had minimal side-channel leakage, so AES used extra defenses to prevent leakage.	Threats: Power analysis and timing. Mitigation: Twofish is certainly resistant; AES requires secure implementation.
Key Management	Moderate: Increases management difficulty due to the complex key schedule	High: Effective and broadly supported key management	Key management assessments increase Twofish's complexity, whereas AES promotes simpler key schedules and larger tool support.	Threats: Short rotation, key compromise. Mitigation: Robust key management, periodic rotation, and secure storage are implemented.
Encryption/Decryption Speed	Moderate: Slower due to complicated operations	High: Faster due to the optimized framework	In practice, AES outperformed Twofish in both encryption and decryption speeds.	Threats: Performance bottlenecks. Mitigation: Use AES for speed, Twofish for security, and increase RAM size to enhance performance.
Resistance to Known Attacks	High: No practical attacks originate	High: No practical attacks originate	Both algorithms passed various known attack methods (e.g., differential, linear cryptanalysis), but none of them were effective.	Threats: Known cryptanalytic methods. Mitigation: Both strategies offer strong protection against such attacks.
Scalability in Distributed Systems	High: Scale fine in distributed environments	High: Efficient in distributed systems	Integration exams ensure the efficient use of both algorithms in the MRF.	Threats: Integration difficulty. Mitigation: Ensure complete integration through full testing and optimization.
Overall Security Level	High: Strong in resisting numerous attacks	High: Fast with moderate security features	Complete testing confirmed that combining both algorithms provides a balanced approach, enhancing security without significantly compromising performance.	Threats: Multi-vector attacks combining different methods. Mitigation: Twofish and AES layering provide a defense-in-depth approach.

Table 7 presents a comparative study of Twofish and AES based on various factors, such as cryptographic strength, resistance to attack, performance, and scalability. The results in Table 7 show how the suggested HE offers a comprehensive approach to securing HDFS data and provides security mechanisms to defend against side-channel and cryptanalysis attacks. Both Twofish and AES have strong cryptographic strengths to resist brute-force and differential analysis attacks. This confirms that the suggested HE remains robust in a distributed environment with layered protection. Twofish is resilient against side-channel attacks owing to its structure, whereas AES requires careful implementation to avoid threats. This dissimilarity enforces the Twofish in the first layer to protect the data with higher openness. In the key management case, AES is a simple key management that improves operational efficiency [31], and Twofish adds entropy and diversity to the encryption process owing to its complex key management infrastructure. This suggests that the HE approach uses the PKI/TLS protocol to integrate robust key management and secure key distribution to ensure a minimum key compromise risk. AES has an optimized framework and is fast due to its hardware acceleration structure, and Twofish is offset by allocating sufficient RAM and leveraging Hadoop MRF processing. Thus, the proposed HE approach provides encryption efficiency and superior security. They have been carefully implemented in the HDFS environment. From a scalability perspective, both algorithms are easily integrated into the MRF framework for parallel processing. This scheme provides a strong, scalable, and secure encryption method with a defense-in-depth strategy that does not compromise the operational efficiency of big data.

3.10.1. Security Margin Comparison

Table 8: Comparison-Based Security Margin

Metric	Twofish	AES	Hybrid
Key space	2^{256}	2^{256}	2^{256}(Sequentially)
Best Known Attack	None	None	Double-layered protection
Side channel attack risk	Low	moderate	Mitigate via GSM
Cryptanalysis attack resistance	High	high	Highest

The security margin comparison based on various factors, such as key space, known attacks, side-channel attacks, and cryptanalysis attacks, is shown in Table 8. In this suggested HE approach, Twofish and AES integration used 2^{256} (sequentially) keys. In the suggested HE, the best-known attack is negligible because of double-layer protection, and side-channel attacks are mitigated because of the GSM mode used by AES-256. This suggested HE approach also resists cryptanalytic attacks. Thus, Table 8 shows that the double-layered protection and MRF framework are suitable for handling large datasets.

3.11. Flow Diagram of A Hybrid Encryption Scheme

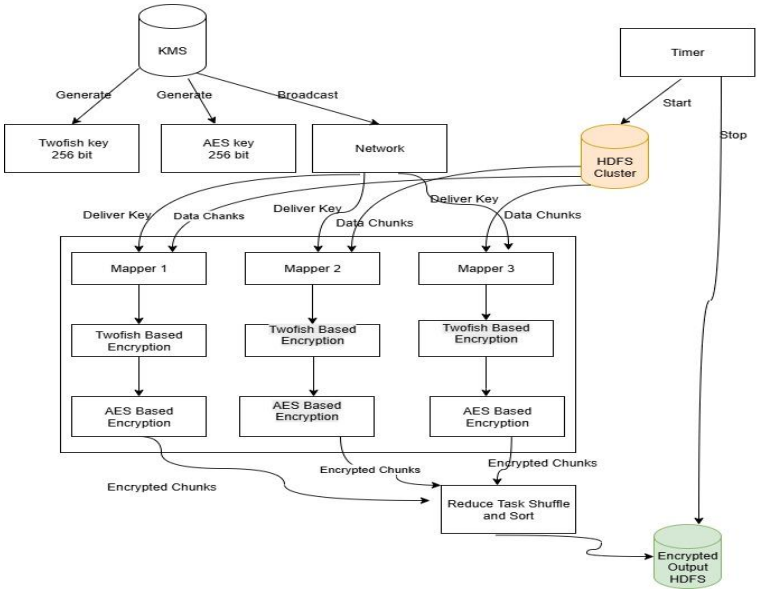

Fig. 5: Flow Diagram of a Hybrid Encryption Scheme.

The suggested HE Scheme, shown in Fig. 5, provides a dual layer of protection for massive datasets stored in the Hadoop framework. First, the data are divided into data chunks of 128 MB HDFS blocks, and each 128 MB block is processed using a mapper. The mapper performs dual-level encryption to encrypt the data. The first layer encrypts the data chunks using the Twofish encryption method, and the second layer encrypts the data from the first layer using the AES-GSM mode. (key management server generates the Twofish (256-bit) and AES keys (256-bit) and broadcasts them to the network, which then delivers the key (512-bit) to each mapper. Subsequently, each mapper encrypts the data and provides the encrypted chunk output to the reducer. The reducer takes the encrypted chunks output and performs shuffle, sort, and aggregate processing. Finally, the reducer provides the encrypted output again to the HDFS. Finally, the HDFS stores the encrypted output. Thus, this study offers a dual level of encryption to fortify data in HDFS. A single level of encryption, such as AES, is not capable of securing HDFS data because of different attacks and small key-size issues [32]. The hybrid method is used to integrate Twofish with AES to improve security and overcome the limitations of AES [29]. Twofish is difficult to decode for an attacker because of its complex structure. The proposed model provides a robust security mechanism for data protection.

3.12. Pseudocode of Proposed Encryption Scheme

```
    #Generate Keys
    Func gen_key ():
    master_key= RANDOM_BYTES(32)
    salt= RANDOM_BYTES(16)
    two_key= SCRYPT(master_key,salt,length=32)
    aes_key=SCRYPT(master_key, salt+"aes",length=32)
    RETURN (two_key,aes_key)
    # Hybrid Encryption Process
    Funtction hybrid_encrypt(data, two_key, aes_key):
    //Layer 1: Twofish Encryption
    tfish_Nonce= RANDOM_BYTES(16)
    tfish_cipher=Twofish(two_key)
    tfish_ciphertext, tfish_tag=tfish_Encrypt_AND_DIGEST(data)
    aes_nonce=RANDOM_BYTES(12)
    aes_cipher=(aes_key,MODE_GCM,nonce=aes_nonce)
    aes_ciphertext, aes_tag=aes_cipher.ENCRYPT_AND_DIGEST(tfish_ciphertext)
    RETURN tfish_nonce+tfish_tag+aes_tag+aes_ciphertext
    #MapReduce Implementation
// Mapper Function
FUNCTION mapper(block_id, block_data):
 keys = GET_BROADCASTED_KEYS()
encrypted_chunks = []
FOR i FROM 0 TO block_data.length STEP CHUNK_SIZE:
chunk = block_data[i:i+CHUNK_SIZE]
encrypted = hybrid_encrypt(chunk, keys.two_key, keys.aes_key)
encrypted_chunks.APPEND((block_id, i, encrypted))
RETURN encrypted_chunks
// Reducer Function
FUNCTION reducer (block_id, encrypted_parts):
 sorted_parts = SORT_BY_OFFSET (encrypted_parts)
 combined = CONCATENATE_ALL (sorted_parts)
RETURN (block_id, combined)
```

```
// Driver Program
MAIN:
// Initialize
 spark_conf = SET_CONFIG (
 blocksize=128MB,
executors=4,
cores_per_executor=2
 )
 sc = SPARK_CONTEXT(spark_conf)
keys = generate_keys()
BROADCAST_KEYS(keys)
 // Job Execution
input_rdd = LOAD_HDFS_DATA("hdfs://path/to/data")
encrypted_rdd = input_rdd.MAP(mapper).REDUCE_BY_KEY(reducer)
 // Output
       SAVE_TO_HDFS (encrypted_rdd, "hdfs://output/path")
       # Decryption Process
FUNCTION hybrid_decrypt(encrypted_data, two_key, aes_key):
// Parse components
 tfish_nonce = encrypted_data[0:16]
 tfish_tag = encrypted_data[16:32]
 aes_nonce = encrypted_data[32:44]
 aes_tag = encrypted_data[44:60]
 ciphertext = encrypted_data[60:]
 // Layer 2: AES-GCM
 aes_cipher = AES(aes_key, MODE_GCM, nonce=aes_nonce)
 tfish_encrypted= aes_cipher.DECRYPT_AND_VERIFY(ciphertext, aes_tag)
// Layer 1: Twofish-EAX
 tfish_cipher = Twofish(two_key)
 RETURN tfish_cipher.DECRYPT_AND_VERIFY(tf_encrypted, tf_tag)
```

4. Results and Discussion

In this study, an experiment was conducted on Google Colab using the JDK 1.8 version and the Hadoop 3 environment. Pyspark was used for the implementation and testing of large datasets (32 MB-256 MB). This test takes input data as plain text and adjusts it to the file size required for constant repetition. We measured the performance of the proposed HE method with an MRF in terms of speed, efficiency, and throughput.

This section presents and analyzes the results obtained from implementing the proposed hybrid encryption (HE) scheme using Twofish and AES within a MapReduce Framework (MRF). Key performance metrics, such as the encryption time /decryption time, efficiency, and throughput, are discussed. The results were benchmarked against the data size (MB) with CP-ABE + AES [30], AES+RSA [9], AES [9], and Twofish [38] encryption times (min), followed by discussions on the statistical validation based on the results, real-world applicability and feasibility of the proposed HE approach to the real world, its integration challenges, limitations, mitigation strategies, and scalability. The MRF uses the concepts of mapper and reducer. MRF performance depends on adjusting memory and CPU allocation. The data processing engine can perform data splitting, inputting, and outputting tasks. The runtime environment of MapReduce supports program operation.

4.1. Comparison of Encryption and Decryption Times between The Proposed HE and other HE Schemes, Alongside A Standalone Encryption Scheme

The encryption time is determined from the time taken to generate the plaintext from the ciphertext. To compare the performance based on the encryption time (min) of the suggested HE model, CP-ABE + AES [30], AES+RSA [9], AES [9], and Twofish [38] encryption methods for data size (MB). The plain text data size was 32 MB-256 MB accordingly to the test setup. In summary, the execution times of the proposed HE method, standalone encryption method, and other hybrid encryption methods were obtained (Fig. 6). Fig. 6 and Table 9 show that the suggested HE provides a minimum encryption time to encrypt all files. The encryption time increased gradually from small to large files, that is, from 32 to 256 MB in size. The encryption time of the proposed HE is 0.03 minutes based on a 32 MB file size. The 256 MB file size also takes 0.2 minutes for encryption. The standalone encryption methods AES and Twofish take 0.67 and 0.28 minutes for encrypting the plaintext data size (32 MB), respectively, and 2.8 and 2.3 minutes for 256 MB, respectively. Similarly, CP-ABE+AES and AES+RSA take a 32 MB file and encrypt the file in 0.93 and 0.72 minutes, respectively. In addition, a 256 MB file was encrypted in 2.82 min and 2.5 min. The results demonstrate that the encryption time required to encrypt the plaintext of the proposed method is highly impressive.

The decryption durations for the proposed HE method, standalone encryption method, and other hybrid encryption techniques are shown in Fig. 7. According to Fig. 7 and Table 10. The proposed HE method achieves the shortest decryption time for all files. As the file size increases from 32 to 256 MB, the decryption time gradually increases. For a 32 MB data size, the proposed HE method requires 0.009 minutes for decryption, while a 256 MB file takes 0.08 minutes. In contrast, standalone encryption methods, such as AES and Twofish, require 1.2 and 0.21 minutes, respectively, to decrypt a 32 MB ciphertext, and 1.3 and 0.49 minutes for a 256 MB file. Similarly, CP-ABE+AES and AES+RSA decrypt a 32 MB file in 0.8 and 1.17 min, respectively, and a 256 MB file in 2.5 and 2.86 min. These findings highlight the impressive effectiveness of the proposed method's decryption time for the ciphertext.

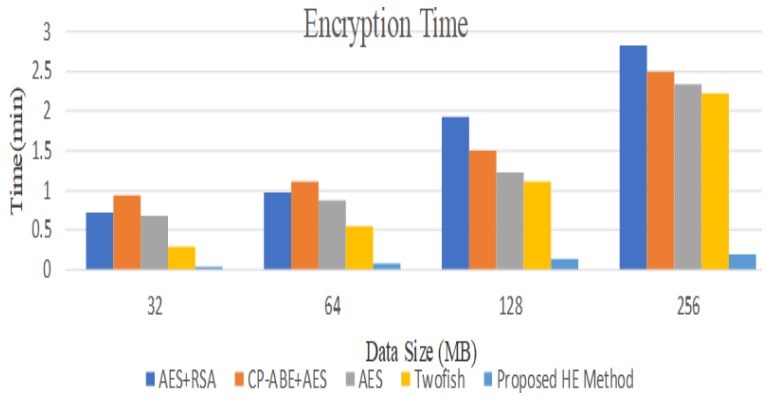

Fig. 6: Encryption Time Comparison between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish.

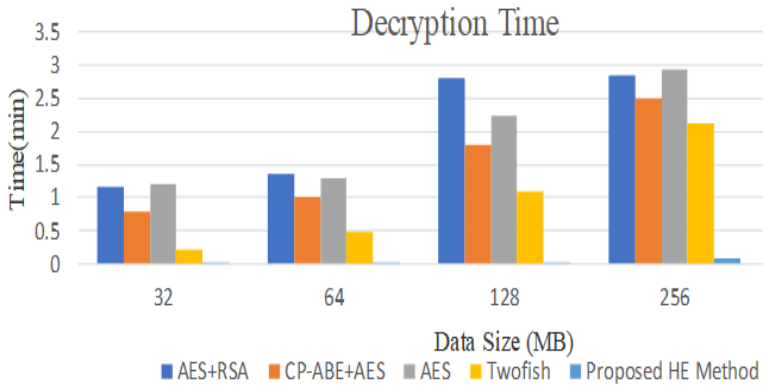

Fig. 7: Decryption Time Comparison between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish.

Table 9: Comparison of Encryption Times between the Suggested HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish

Data Size (MB)	AES+RSA (min)	CP-ABE+AES (min)	AES (min)	Twofish (min)	Proposed HE Method (min)
32	0.72	0.93	0.67	0.28	0.03
64	0.98	1.1	0.87	0.55	0.08
128	1.93	1.5	1.22	1.11	0.13
256	2.82	2.5	2.33	2.21	0.2

Table 10: Comparison of Decryption Times between the Suggested HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish

Data Size (MB)	AES+RSA (min)	CP-ABE+AES (min)	AES (min)	Twofish (min)	Proposed HE Method (min)
32	1.17	0.8	1.2	0.21	0.009
64	1.37	1	1.3	0.49	0.02
128	2.8	1.8	2.23	1.09	0.03
256	2.86	2.5	2.93	2.12	0.08

4.2. The Encryption and Decryption Process Performance Changes with The Amount of Data Size

This test evaluates the encryption process performance (efficiency) in terms of the plaintext data size and the running time of the proposed algorithm. A comparison of the efficiency of other encryption methods, such as AES, Twofish, CP-ABE+AES, AES+RSA, and the proposed HE, is shown in Fig. 8 and Table 11. The proposed HE method efficiently handles large datasets with minimal latency. The results demonstrate that the more encrypted the data, the greater is the efficiency. In addition, Fig. 8 shows that with an increase in the plaintext size, the encryption efficiency improves. Therefore, this encryption scheme is well-suited for this purpose. The encryption time-based efficiency of the proposed HE approach for 32 and 256 MB of data was 1066.7 and 1280 MB/min, respectively. The results show that the proposed HE approach provides better results than the existing approaches.

The performance of the decryption process in terms of data size and execution time for the proposed algorithm to decrypt the ciphertext was evaluated. Fig. 9, and Table 12. presents a comparison of its efficiency with other encryption techniques, including AES, Twofish, CP-ABE+AES, AES+RSA, and the proposed HE. The proposed HE method can manage both large data volumes and minimal waiting periods. Furthermore, Fig. 9 illustrates that as the size of the ciphertext increases, the decryption efficiency increases. The efficiency of the proposed HE method, based on the decryption time, was 3555.5 and 3200 MB/min for 32 and 256 MB of data, respectively. The findings indicate that the proposed HE method with MRF outperforms existing encryption techniques.

The encryption Efficiency formula is given as

$$\text{Efficiency (MB/min)} = \text{Plaintext Data} / \text{Encryption Time} \quad (1)$$

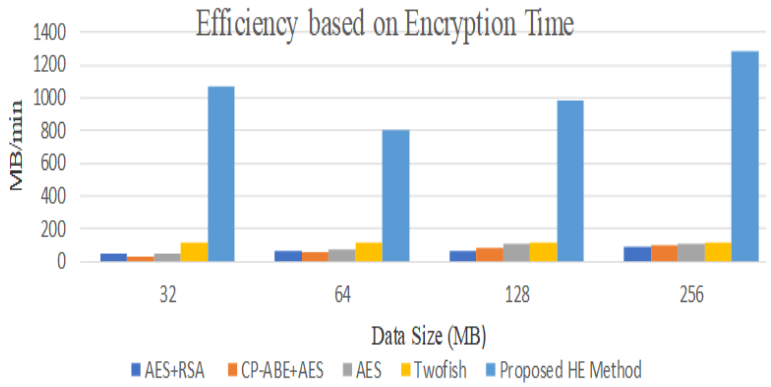

Fig. 8: Encryption Time-Based Efficiency Comparison between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish.

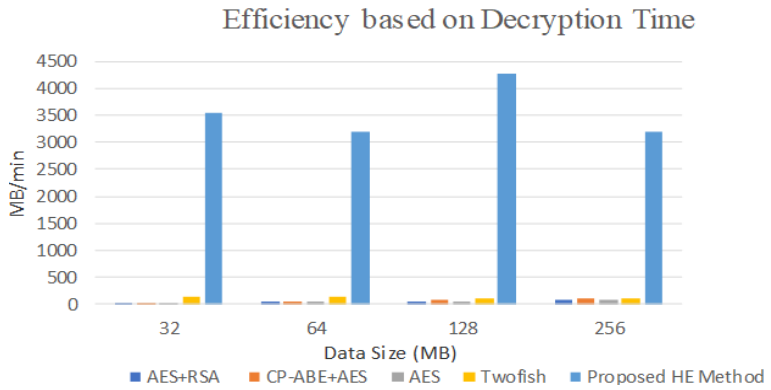

Fig. 9: Decryption Time-Based Efficiency Comparison between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish.

Table 11: Comparison of Efficiency in Encryption Time and Throughput Time between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish

Data Size (MB)	AES+RSA (MB/min)	CP-ABE+AES (MB/min)	AES (MB/min)	Twofish (MB/min)	Proposed HE Method (MB/min)
32	44.44444	34.4086	47.76119	114.2857	1066.667
64	65.30612	58.18182	73.56322	116.3636	800
128	66.32124	85.33333	104.918	115.3153	984.6154
256	90.78014	102.4	109.8712	115.8371	1280
Throughput (MB/min)	74.4186	79.60199	94.30255	115.6627	1090.909091

Table 12: Comparison of Decryption Time Efficiency and Throughput Time between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish

Data Size (MB)	AES+RSA (MB/min)	CP-ABE+AES (MB/min)	AES (MB/min)	Twofish (MB/min)	Proposed HE Method (MB/min)
32	27.35043	40	26.66667	152.381	3555.556
64	46.71533	64	49.23077	130.6122	3200
128	45.71429	71.11111	57.3991	117.4312	4266.667
256	89.51049	102.4	87.37201	120.7547	3200
Throughput (MB/Min)	58.53659	78.68852	62.66319	122.7621	3453.237

4.3. Throughput Time of The Proposed HE Versus The Standard HE Schemes and The Hybrid Encryption Scheme

When calculating the throughput, the encryption time is a factor related to speed. For the encryption technique, the throughput is determined by dividing the total amount of plaintext data by the total encryption time. Fig. 10, and Table 11. shows the throughput time of the encryption process of the proposed HE method compared with the conventional and standalone hybrid encryption methods. Also, Fig. 11 and Table 12. shows the throughput time of the proposed HE decryption process with the conventional hybrid and standalone encryption methods. The proposed HE method encryption and decryption throughput times were 1093.2 and 3553.2 MB/min, respectively. The results show that the proposed HE encryption approach is highly effective.
Throughput of the proposed encryption algorithm

(Megabytes/min) = Total size of plaintext / Total time of Encryption (2)

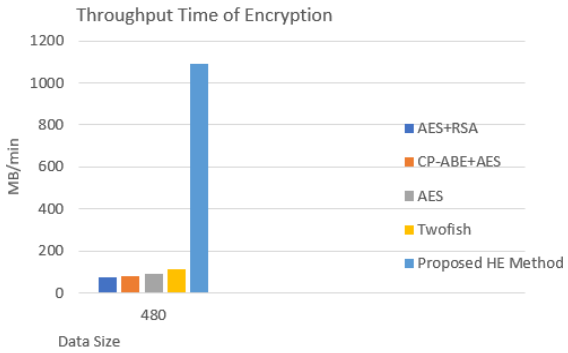

Fig. 10: Throughput Time of Encryption-Based Comparison between the Proposed HE Scheme and the AES+RSA, CP-ABE+AES, AES, and Twofish.

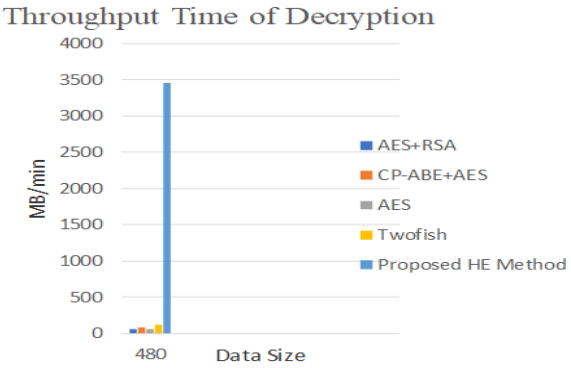

Fig. 11: Throughput Time of Decryption Based Comparison between the Proposed HE Scheme and AES+RSA, CP-ABE+AES, AES, and Twofish

Fig. 6 to Fig. 11 illustrate that while AES is generally effective, it encounters several challenges and limitations when safeguarding HDFS data in cloud environments (D'souza & Panchal, 2017). This study integrates Twofish with AES and MRF for parallel processing capability to overcome AES limitations and those of other hybrid encryption approaches. Therefore, compared with standard and other hybrid encryption methods, the proposed approach provides robust security and enhanced performance. In summary, the experimental results validate that the proposed hybrid encryption scheme with MRF capabilities achieves notable improvements in speed, efficiency, and throughput compared with the standard and other hybrid encryption models. These gains, coupled with their adaptability to parallel environments via MRF, make them suitable for security-critical and high-volume applications across various industries.

4.4. Statistical Validation based on The Result

The mean and standard deviation based on the encryption and decryption times of the proposed HE approach, AES+RSA, CP-ABE+AES, AES, and Twofish are calculated and presented in Table 13. The table below highlights that the proposed HE approach based on Twofish-256 and AES-256 provides better results than standalone encryption, AES, Twofish, and other hybrid encryption methods, AES+RSA, and CP-ABE+AES.

The encryption time-based ANOVA test gives an f-value of 2.67 and a p-value of 0.07. In addition, the decryption time based on the Anova test based on f and p values is 9.9 and 0.0003, respectively. The ANOVA test shows that the proposed HE approach is credible. The proposed HE approach has a minimum average encryption time of 0.11 minutes, which is acceptable. This variance is not significant at the 95% confidence value (f=2.67, p=0.07) owing to the small sample size; however, the decryption time shows a high variance (f=9.9 and p<0.001) based on the minimum mean value and p value. This statistical validation shows that the suggested approach is practically feasible for a time-intensive decryption operation.

The Mean is calculated by

$$\text{Mean} = \frac{\sum x_i}{n} \tag{3}$$

Where x_i is the encryption time value, and n is the total data size.

The standard deviation calculated by the

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})2}{n-1}} \tag{4}$$

Where X_i and $\overline{X}$ is the encryption time and the encryption mean.

Table 13: Statistical Validation of the Suggested HE Approach to Other Encryption Approaches to Secure Data

Method	Encryption Time-based Mean (Min)	Decryption Time-based Mean (Min)	Encryption Time-based SD(Min)	Decryption Time-based SD (Min)
AES+RSA	1.6	1.9	0.9	0.7
CP-ABE+AES	1.5	1.5	0.7	0.7
AES	1.2	1.9	0.7	0.6
Twofish	1.03	0.3	0.8	0.1
Proposed HE approach	0.11	0.03	0.07	0.03

The ANOVA encryption time (F=2.67, p=0.07) result falls short of standard significance levels (α=0.05). This is due to the limited sample size in our experiments. Future work will validate these findings using expanded datasets (e.g., 512 MB–1 GB) and increased trial repetitions to enhance statistical power. But the scheme's throughput superiority over benchmarks (Table 3) and compliance with critical regulations (Table 4) underscore its real-world feasibility.

4.5. Real-world Applicability

The suggested HE is widely adopted in the industry to protect highly sensitive data from unauthorized access. In practice, this approach is effective in industries that require robust security measures and rapid processing of big data. Now, let me explain how robust this approach is for application in real-world scenarios.

4.5.1. The Healthcare Industry (HIPAA Compliance)

The proposed method is suitable for the Health Insurance Portability and Accountability Act (HIPAA) and provides secure storage of electronic protected health information (ePHI). In the healthcare sector, there is a need to maintain the integrity, confidentiality, and availability of data in real-time operations. Twofish is known for its resistance to side-channel attacks, such as timing and power analysis, which are critical when handling patient data. Combined with AES efficiency, this approach enables large data processing with minimal latency. This is essential when telemedicine, real-time diagnosis, and cloud-hosted patient management are necessary.

4.5.2. Financial Sector (GDPR Compliance)

The financial industry is a perfect example of where the protection of personal data (PII) under the GDPR is needed, and this hybrid solution offers such protection to secure financial records and PII against data breaches. Twofish features ensure that sensitive financial data is protected from internal and external threats, whereas AES's fast processing capabilities enable real-time reporting and data analysis. Together, both encryption methods provide a highly scalable and secure encryption solution.

4.5.3. Government and Defense

Governments and defense deal with highly sensitive information that is often targeted by APT. The defense-in-depth model of the proposed HE method suggests lower vulnerability to APT and zero-day exploits. This provides high security against advanced cyber threats to government departments and defense organizations. Secret data in transit and at rest use multilayered encryption, enhancing protection against data exfiltration and surveillance ([NIST SP 800-207]). This approach is mainly suited for securing mission-critical systems.

4.5.4. Big Data Analytics

Secure large-scale data can be processed rapidly using a parallel processing framework (MRF). This Twofish with AES combination has hardware acceleration and robust security features that guard sensitive data, making it particularly well-suited for Big Data applications in areas such as retailing, telecommunications, and energy.
The suggested HE, Twofish, and AES encryption techniques with MRF are ideal for many industries in which high security and efficient data processing are required. A unique transmission of data in its vital areas, including healthcare, finance, government, Big Data analytics, and cloud storage, provides security by offering regulatory compliance and cyber threat resistance. This suggests HE makes it an ideal solution for cases where the chances of cyber-attacks are high. This method ensures that the data remains secure and scalable while maintaining performance.

4.6. Real-World Deployment Feasibility and Integration Challenges

This suggested HE approach is designed for practical deployment within the Hadoop 3.x distribution without any structural changes. The following key points are mentioned in terms of deployment feasibility.

4.6.1. Compatible with MRF (Map-Reduce Framework)

The proposed approach was implemented in the Hadoop MapReduce framework for parallel processing. Both algorithms are symmetric encryption, and the Python library is easy to deploy in the existing structure.

4.6.2. Scalable Parallel Processing

The encryption and decryption tasks are distributed through the mapper and reducer through the parallel processing capability in the Hadoop environment. Thus, this approach maintains dual-level encryption with scalability across the distributed data blocks.

4.6.3. Key Management Integration

Key management servers (KMSs) seamlessly integrate with the PKI/TLS protocol to facilitate key distribution in each mapper for encryption and decryption. Consequently, this architecture is well-suited for key rotation, revocation, and distribution. The keys for Twofish and AES were generated using a pseudocode generator to ensure high entropy and unpredictability. These keys are stored in an encrypted repository, such as a secure software vault, to prevent unauthorized access. The Hadoop framework employs Transmission Layer Security (TLS) to securely transmit keys between the KMS and authorized data nodes. Fine-grained access policies determine which nodes and services can request keys, thereby minimizing the risk of insider threats and unauthorized key retrieval. The KMS oversees the entire key lifecycle. The integration of Apache KMS ensures that the processes of generation, distribution, rotation, and revocation adhere to industry-standard security protocols. Keys are rotated according to policy to limit exposure in case of compromise. Rotation occurs seamlessly, with the old key being used to decrypt existing data and a new encryption key being applied to incoming data. During the key revocation phase, if the KMS suspects that a key has been compromised, it can revoke it, rendering all future decryption attempts with that key invalid. Re-encryption is then performed using a new key. Apache Ranger employs role-based access control and attribute-based access control policies to ensure that only authorized users can request, access, and manage keys. All key operations were logged for traceability and compliance.

Keys are stored in secure vaults to prevent unauthorized access, and secure communication channels are used to provide keys to data nodes, protecting against man-in-the-middle attacks. This framework for the key management lifecycle enhances the security of the proposed HE approach, making it resilient against key compromises. This key lifecycle management ensures smooth operation and compliance with security rules.

4.7. Integration Challenges, Mitigation Strategies, and Scalability Discussion

Despite the adoption of feasibility measures, several integration issues can arise, including computational overhead, memory I/O strain, KMS vulnerabilities, increased ciphertext size, and the need to counter side-channel attacks (SCA). In terms of computational overhead, double-layered encryption results in higher CPU consumption for data nodes with limited resource availability. We addressed this issue by employing hardware acceleration and batch processing. For memory and I/O strain, Twofish requires more memory, which we addressed by adjusting the mapper block size. The KMS is susceptible to failure; hence, we mitigate this by deploying replicated KMS nodes and integrating them with Zookeeper for failure management. The increase in the ciphertext size is due to the expansion of the cipher from the double-layered encryption nonce and the tag. To address this, compression and combined tags were used to mitigate this, and throughput should be improved, and reprocessing minimized, when security checks fail. Finally, in terms of mitigating side-channel attacks, Twofish is naturally resistant to such vulnerabilities, whereas AES must be implemented carefully, and the AES-GSM library must be used to avoid potential weaknesses. The proposed HE method ensures that both encryption systems remain secure within the MRF, preventing sensitive information leakage. In conclusion, this is a recommended HE approach that can be smoothly integrated into the Hadoop framework using cluster management tools. Another challenge is scalability in extreme scenarios, particularly when MRF is used for the parallel processing of large datasets. In cases involving extremely large datasets, such as those in the petabyte and exabyte ranges, this suggests that HE has not been extensively evaluated. As the data size increases exponentially, the efficiency gains diminish. Therefore, alternative platforms, such as Spark, Flink, and cloud solutions, are employed to handle massive data volumes.

The suggested HE method shows a strong performance for datasets reaching several terabytes. Nonetheless, when scaling workloads at the petabyte or exabyte level within HDFS, complexities arise that must be addressed to ensure both security and efficiency.

In large-scale clusters, the encryption and decryption processes generate substantial network loads, mainly because of the shuffling operation in MapReduce tasks, which can lead to longer job completion times. To mitigate this, strategies such as deploying ultra-high-bandwidth infrastructure and advanced data locality scheduling are recommended to reduce internode data transfer.

It is challenging to distribute keys securely across thousands of nodes. A highly available KMS, spread across multiple nodes and paired with edge caching of session keys and lightweight key derivation functions, can help decrease latency in extensive data environments.

Differences in node capabilities, such as CPU performance, available memory, and hardware acceleration support, can cause performance variations during encryption and decryption. Adaptive encryption scheduling and workload balancing are essential for optimizing throughput on various platforms.

Node failures become more common as clusters grow. In such cases, encrypted data recovery must ensure continuous access to valid keys, which is achieved through KMS replication, regular key backups, and automated re-keys for insufficient datasets.

Future optimization strategies will involve integrating high-performance frameworks and using hardware accelerators for regional key caching. Additionally, hybrid cloud-HDFS models with edge computing nodes are anticipated to offload encryption tasks closer to the data sources, thus alleviating central processing bottlenecks.

By thoroughly addressing these scalability challenges, the proposed HE framework can be developed into an exabyte-ready solution, maintaining performance, security, and compliance across new big data systems.

5. Conclusion and Future Work

This research sheds light on the encryption of large-scale data on cloud platforms by employing the MRF and the newly developed HE Twofish alongside AES. To implement this method, it is essential to first establish a Hadoop framework and then integrate the proposed HE. This dual encryption strategy addresses the limitations of using AES alone, as AES is not well-suited to cloud environments. Twofish provides robust data security, whereas AES ensures rapid encryption and decryption, making their combination advantageous in terms of both security and performance. Furthermore, the effectiveness of the proposed HE method with MapReduce was demonstrated, and its feasibility was confirmed by encrypting a significant amount of plaintext data. This study examined the practicality of the proposed HE model. In addition, the proposed HE scheme was evaluated against other traditional HE and standalone encryption methods. Compared with standalone (AES, Twofish) and other hybrid encryption methods (AES+RSA, CP-ABE + AES, AES+RSA), the proposed HE scheme enhances the encryption efficiency by over 6% (MB/min), encryption speed by more than 5-6%, and throughput by over 6% (MB/min). Moreover, statistical validation using an ANOVA test supports the credibility of the proposed approach in terms of encryption and decryption values, with an f-value of 2.67 and p-value of 0.07, and f=9.9 and $p<0.001$. These test results indicate that the proposed approach is highly significant in big data environments. In conclusion, the experimental findings confirm that the proposed hybrid encryption scheme significantly enhances speed, efficiency, and throughput compared to standard encryption models and other hybrid encryption methods. Additionally, this proposed HE approach provides depth-in-security and aligns with regulatory compliance standards and their sectors. These improvements, along with their compatibility with parallel environments via the MRF, make them ideal for security-sensitive high-volume applications across various sectors. Future research could explore the potential of integrating hardware-based security measures, such as a Trusted Platform Module, to further enhance the reliability and security of the HDFS storage platform.

Acknowledgment

We thank the publisher for their support in publishing this research. We are grateful for the resources and platforms provided by the publisher that have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team for reviewing and editing our work and are thankful for the opportunity to contribute to the field of research through this publication.

References

[1] Bertino E, Ferrari E. (2017). Big data security and privacy. In: Studies in big data (pp. 425–439). Available from: https://doi.org/10.1007/978-3-319-61893-7_25.

[2] Warkentin M, Orgeron C. (2020). Using the security triad to assess blockchain technology in public sector applications. International Journal of Information Management. https://doi.org/10.1016/j.ijinfomgt.2020.102090.

[3] Yang P, Xiong N, Ren J. (2020). Data Security and Privacy Protection for Cloud Storage: a survey. IEEE Access, 8,131723–40. Available from: https://doi.org/10.1109/ACCESS.2020.3009876.

[4] Narayanan A, Toubiana V, Barocas S, Nissenbaum H, Boneh D. A critical look at decentralized personal data architectures. arXiv (Cornell University). https://arxiv.org/abs/1202.4503.

[5] White T. Hadoop: The Definitive Guide. "O'Reilly Media, Inc.".

[6] White T. (2015) Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale. "O'Reilly Media, Inc.".

[7] Borthakur, D. (2007). The Hadoop Distributed File System: Architecture and Design (vol 1–14).

[8] Filaly Y, Mendili FE, Berros N, Idrissi Y.E.B.E. (2023). Hybrid Encryption Algorithm for Information Security in Hadoop. International Journal of Advanced Computer Science & Applications, 14(6). https://doi.org/10.14569/IJACSA.2023.01406137.

[9] Sunder A, Shabu N, Nair TR. (2021). Securing big data in Hadoop using hybrid encryption. In: Smart innovation, systems and technologies. p. 521–30. Available from: https://doi.org/10.1007/978-981-16-3675-2_39.

[10] Viswanath G, Krishna PV. (2020). Hybrid encryption framework for securing big data storage in multi-cloud environment. Evolutionary Intelligence, 14(2),691–698. https://doi.org/10.1007/s12065-020-00404-w.

[11] Negi K, Shrestha R, Borges TL, Sahana S, Das S. (2023). A hybrid cryptographic approach for secure Cloud-Based file storage. https://doi.org/10.1109/GlobConET56651.2023.10150148.

[12] Lai JF, Heng SH. (2022). Secure File Storage on Cloud Using Hybrid Cryptography. Journal of Informatics and Web Engineering, 1(2):1–18. Available from: https://doi.org/10.33093/jiwe.2022.1.2.1.

[13] Kumari N, Malhotra V. (2022). Secure cloud data storage using hybrid cryptography. International Journal for Research in Applied Science and Engineering Technology, 10(4), 60–63. https://doi.org/10.22214/ijraset.2022.41081.

[14] Chaudhari A. (2023). A survey on hybrid cryptography for secure file storage on the cloud. International Journal for Research in Applied Science and Engineering Technology, 11(6):2523–2525. https://doi.org/10.22214/ijraset.2023.54089.

[15] Jain P, Gyanchandani M, Khare N. (2019). Enhanced Secured Map Reduce layer for Big Data privacy and security. Journal of Big Data, 6(1). https://doi.org/10.1186/s40537-019-0193-4.

[16] Gupta M, Dwivedi RK. (2023). Fortified MapReduce Layer: Elevating security and privacy in big data. ICST Transactions on Scalable Information Systems. https://doi.org/10.4108/eetsis.3859

[17] Bangera S, Billava P, Naik S. (2020). A Hybrid Encryption Approach for Secured Authentication and Enhancement in Confidentiality of Data. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-000145

[18] Jintcharadze E, Iavich M. (2020). Hybrid Implementation of Twofish, AES, ElGamal and RSA Cryptosystems. https://doi.org/10.1109/EWDTS50664.2020.9224901.

[19] Schneier B. (2005). "Twofish Cryptanalysis Rumors." Schneier on Security Blog.

[20] NIST announces Encryption Standard finalists. 2017. Available from: https://www.nist.gov/news-events/news/1999/08/nist-announces-encryption-standard-finalists.

[21] Menezes AJ, Van Oorschot PC, Vanstone SA. HANDBOOK of APPLIED CRYPTOGRAPHY. https://theswissbay.ch/pdf/Gentoomen%20Library/Cryptography/Handbook/Menezes.pdf.

[22] Yan X, Zhu Z, Wu Q. (2018). Intelligent inversion method for pre-stack seismic big data based on MapReduce. Computers & Geosciences, 110, 81–89. https://doi.org/10.1016/j.cageo.2017.10.002.

[23] Rahim LA, Kudiri KM, Bhattacharjee S. Framework for parallelization on big data. PloS One, 14(5), e0214044. https://doi.org/10.1371/journal.pone.0214044.

[24] Khan M, Jin Y, Li M, Xiang Y, Jiang C. (2015). Hadoop performance modeling for job estimation and resource provisioning. IEEE Transactions on Parallel and Distributed Systems. 27(2),441–454. https://doi.org/10.1109/TPDS.2015.2405552

[25] Ma C, Zhao M, Zhao Y. (2023). An overview of Hadoop applications in transportation big data. Journal of Traffic and Transportation Engineering/Journal of Traffic and Transportation Engineering.10(5),900–917. https://doi.org/10.1016/j.jtte.2023.05.003

[26] Taylor RC. (2010). An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics.11(S12). Available from: https://doi.org/10.1186/1471-2105-11-S12-S1.

[27] Vohra D. (2016). Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools. Apress. https://doi.org/10.1007/978-1-4842-2199-0.

[28] Ahmed N, Barczak ALC, Susnjak T, Rashid MA. (2020). A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. Journal of Big Data, 7(1). https://doi.org/10.1186/s40537-020-00388-5.

[29] Al, PS. (2021). Performance analysis of cascaded Hybrid symmetric encryption models. Türk Bilgisayar Ve Matematik Eğitimi Dergisi, 12(2), 1699–708. https://doi.org/10.17762/turcomat.v12i2.1506.

[30] Mohanraj, T., & R. Santhosh. (2022). Hybrid Encryption Algorithm for Big Data Security in the Hadoop Distributed File System. Computer Assisted Methods in Engineering and Science, 29(1-2),33–48.

[31] Kadre, Viplove, and Sushil Chaturvedi. (2015). AES–MR: A Novel Encryption Scheme for securing Data in HDFS Environment using Map Reduce. International journal of Computer applications, 129.12 ,12-19. https://doi.org/10.5120/ijca2015906994

[32] D'souza, F.J., and D. Panchal. (2017). Advanced encryption standard (AES) security enhancement using hybrid approach (pp. 647-652). https://doi.org/10.1109/CCAA.2017.8229881.

[33] Mahmoud, H., Hegazy, A., and Khafagy, M. H. (2018). An approach for big data security based on Hadoop distributed file system. https://doi.org/10.1109/ITCE.2018.8316608.

[34] Rizvi, S., Hussain, S. Z., and Wadhwa, N. (2011). Performance Analysis of AES and TwoFish Encryption Scheme (pp. 76-79), https://doi.org/10.1109/CSNT.2011.160.

[35] Tian, Y. & Yu, X. (2021). Trustworthiness study of HDFS data storage based on trustworthiness metrics and KMS encryption. 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, pp. 962-966. https://doi.org/10.1109/ICPECA51329.2021.9362537.

[36] Awaysheh, F. M., Aladwan, M. N., & Alazab, M., Alawadi, S., & Cabaleiro, J. C., & Pena, T. F., (2021). Security by Design for Big Data Frameworks Over Cloud Computing. IEEE Transactions on Engineering Management. PP. https://doi.org/10.1109/TEM.2020.3045661

[37] Du, H., Zhang, J., Zhang, J., Sha, S., & Tang, Z. (2019). The Library for Hadoop deflate compression based on FPGA accelerator. 282–287. https://doi.org/10.1109/ComComAp46287.2019.9018820.

[38] Alabdulrazzaq, H., & Alenezi, M. (2022). Performance Analysis and Evaluation of Cryptographic Algorithms: DES, 3DES, Blowfish, Twofish, and Threefish. International Journal of Communication Networks and Information Security (IJCNIS), 14(1). https://doi.org/10.17762/ijcnis.v14i1.5262.

[39] Mothukuri, V., Cheerla, S. S., Parizi, R. M., Zhang, Q., & Choo, K.-K. R. (2021). BlockHDFS: Blockchain-integrated Hadoop distributed file system for secure provenance traceability. 100032. https://doi.org/10.1016/j.bcra.2021.100032.

[40] Zhang, C., Li, Y., Sun, W., & Guan, S. (2020). Blockchain Based Big Data Security Protection Scheme. 574–578. https://doi.org/10.1109/ITOEC49072.2020.9141914.

[41] Xu, C., & Li, J. (2024). Design of intelligent software security system based on SPARK big data Computing. *Wireless Personal Communications*. https://doi.org/10.1007/s11277-024-11015-4.
[42] Liu, J., Liu, Y., & Li, B. (2023). Design and analysis of hash function based on spark and chaos system. International Journal of Network Security, 25(3), 456-467.