Journal of Computer Science

Original Research Paper

Efficient Data Encryption for Securing HDFS Using DQN-Enhanced Deep Reinforcement Learning Techniques

[1]Shivani Awasthi and [2]Narendra Kohli

[1]Department of Computer Science and Engineering, Research Scholar, Harcourt Butler Technical University, Kanpur, India
[2]Department of Computer Science and Engineering, Professor, Harcourt Butler Technical University, Kanpur, India

Article history
Received: 24-10-2024
Revised: 18-11-2024
Accepted: 22-11-2024

Corresponding Author:
Shivani Awasthi
Department of Computer Science and Engineering, Research Scholar, Harcourt Butler Technical University, Kanpur, India
Email: awasthisb@gmail.com

Abstract: In the era of big data, ensuring the security of large-scale distributed storage systems like the Hadoop Distributed File System (HDFS) is critical. Traditional encryption methods often struggle to balance robust security with system performance, leading to vulnerabilities and inefficiencies. This study presents the design and implementation of an efficient data encryption algorithm for securing HDFS using Deep Q-Network (DQN) enhanced Deep Reinforcement Learning (DRL) techniques. The proposed model dynamically optimizes encryption parameters by leveraging the adaptive capabilities of DQN, ensuring robust security while maintaining high system performance and scalability. Our approach addresses key limitations of existing encryption methods by integrating DQN with deep reinforcement learning to create a dynamic encryption framework that adjusts in real time based on data access patterns and threat levels. The results demonstrate significant improvements in security and efficiency compared to conventional encryption techniques. Specifically, the DQN-enhanced DRL algorithm consistently outperformed baseline methods regarding encryption strength, computational efficiency, latency, resource utilization, adaptability, and energy consumption. The contributions of this research include the development of a novel DQN-enhanced encryption algorithm tailored for HDFS, the creation of an adaptive encryption framework that leverages real-time data dynamics, and a thorough evaluation demonstrating the practical benefits of the proposed solution. This study paves the way for future research in intelligent encryption systems, offering a robust and efficient approach to securing large-scale distributed storage environments. Our findings underscore the potential of integrating advanced machine learning techniques into encryption processes to enhance security and performance, addressing the complex challenges modern data storage systems pose.

Keywords: HDFS Security, Data Encryption, Deep Reinforcement Learning, Deep Q-Network, Adaptive Encryption Algorithms

Introduction

In the modern era of big data, securing large-scale distributed storage systems such as the Hadoop Distributed File System (HDFS) has become increasingly critical (Alpaydin, 2020). HDFS, widely used for storing and managing massive amounts of data, faces significant challenges in ensuring data security due to its distributed nature and the sensitive nature of the stored information (Tabbassum *et al.*, 2021). While effective in certain contexts, traditional encryption methods often struggle to balance the trade-offs between robust security and system performance, leading to vulnerabilities and inefficiencies (Alhazmi and Eassa, 2022). This research aims to address these challenges by developing an efficient data encryption algorithm for HDFS utilizing Deep Q-Network (DQN) enhanced Deep Reinforcement Learning (DRL) techniques (Ghemawat *et al.*, 2003)

Rapid advancements in data management technologies have marked the evolution of distributed storage systems Sunder *et al.* (2022). HDFS, a cornerstone of big data platforms like Apache Hadoop, provides scalable and reliable storage by distributing data across multiple nodes Belhadaoui *et al.* (2023). However, the distributed nature of HDFS introduces complexities in maintaining data security, especially as data volumes grow and cyber threats

become more sophisticated (Hamza *et al*., 2022). Traditional encryption methods, such as symmetric and asymmetric encryption, have been the backbone of data security (Al Jallad *et al*., 2019; Khandagale *et al*., 2024). Despite their strengths, these methods often introduce significant computational overhead and latency, impeding system performance and scalability (Mnih *et al*., 2015). The significance of securing HDFS lies in its widespread adoption across various industries, including finance, healthcare, and telecommunications, where data breaches can have severe consequences (Shvachko *et al*., 2010). Historical data breaches have demonstrated the devastating impact of inadequate security measures, leading to financial losses, reputational damage, and legal repercussions (Silver *et al*., 2016). As such, there is a pressing need for advanced encryption solutions that not only enhance security but also maintain system efficiency and adaptability Pronika and Tyagi (2021).

The primary motivation behind this research is to develop a novel encryption algorithm that effectively balances the competing demands of security and performance in HDFS environments (Sood *et al*., 2011). By leveraging the capabilities of DQN-enhanced DRL techniques, this research aims to create an adaptive encryption framework that can dynamically adjust encryption parameters based on real-time data access patterns and threat levels Singh *et al*. (2023). The goal is to enhance the overall security posture of HDFS without compromising on computational efficiency or scalability (Wang *et al*., 2016). Additionally, this research seeks to contribute to the broader field of data security by exploring the potential of reinforcement learning in optimizing encryption processes (Nenov, 2024). By integrating machine learning methodologies with encryption strategies, this study aims to develop a robust, intelligent encryption system that can autonomously respond to evolving threats and data usage patterns. Ultimately, the findings of this research are expected to offer significant insights and practical solutions for securing large-scale distributed storage systems, benefiting both academia and industry (Olaoluwa and Potter, 2024).

In today's digital era, securing large-scale data storage systems like the Hadoop Distributed File System (HDFS) is crucial to protect sensitive information from unauthorized access and cyber threats (Mustafa *et al*., 2020). Existing encryption algorithms often fail to balance security, efficiency, and computational overhead, leading to vulnerabilities and performance bottlenecks (Mashonganyika *et al*., 2020). This research addresses the need for an efficient data encryption algorithm tailored for HDFS by leveraging Deep Q-Network (DQN)--enhanced deep reinforcement learning techniques. The proposed solution aims to optimize encryption processes, ensuring robust security while maintaining high system performance and scalability (Xiao *et al*., 2019) by integrating advanced machine learning methodologies, this study seeks to develop a dynamic, adaptive encryption framework that can intelligently respond to evolving threats and data access patterns, ultimately enhancing the overall security posture of HDFS environments Wang *et al*. (2024).

Addressing the Need for Adaptability and Responsive Security in HDFS

HDFS requires more adaptability and responsive security Measures as listed below.

HDFS Distributed and Dynamic Nature

HDFS used for efficient data storage and retrieval of data across multiple nodes in the distributed network. This setup introduces numerous attacks. Each node and data block can likely attack the surface-making system for more prompt for unauthorized access and beaches. Without adaptive security, standard encryption may fail to protect each node efficiently. When the number of nodes and volume of data grows.

Limitation of Standard Methods

Traditional encryption methods AES and RSA are static and computationally intensive. Traditional encryption methods apply the same method in each block unrelatedly to access frequency and sensitivity. So, the infrequently accessed data is encrypted with a similar computational load as high-priority data, consuming unnecessary resources. Traditional encryption AES and RSA also lack the flexibility to adapt to changing threat points.

Dynamic Threat Environment and Real-Time Adaptation

HDFS environments often involve changing access patterns, varying from high-traffic periods to idle times. In real time an adaptive encryption system, based on DQN-enhanced DRL, could monitor these access forms and threat levels. By doing so, it can dynamically regulate encryption strength and computational resources based on current needs, fortifying data efficiently without imposing unnecessary load during low-activity periods. This reaction is important as data grows. A static model may become increasingly unusable due to the high computational cost of applying uniform encryption across all data nodes, whereas an adaptive model can allocate encryption resources more effectively.

Impacts on Performance and Scalability

Traditional encryption methods can lead to system inefficiencies in HDFS will simplify the need for adaptive security. Traditional methods may cause latency and resource bottlenecks, as HDFS is used in real-time applications that require fast data retrieval and minimal delay. Adaptive security systems can optimize the encryption-decryption cycle.

Potential for Security Breaches

In a static encryption system, attackers who gain access to a node may exploit the uniform encryption parameters across HDFS, increasing the chances of breaching additional nodes. In contrast, an adaptive system can respond to distrustful activity patterns by raising security measures dynamically, thus better protecting sensitive data and reacting promptly to possible cracks.

In this way, HDFS needs adaptive security solutions and DQN-enhanced DRL. This is a powerful solution that traditional methods do not.

The major goal of this research is to design an enhanced HDFS security system utilizing modern deep reinforcement learning approaches as applied to the new advanced encryption algorithm for adaptive data security. It also is the basis for the goal of this research, which is to improve HDFS security while minimizing impacts on system performance and scalability. The specific objectives of this research are as follows:

1. This primer explains the process of developing an enhanced deep reinforcement-learning algorithm based on DQN for improving the parameters of encryption in HDFS.
2. To build a flexible encryption model that can learn and grow as the organization harnesses its data in real-time and faces varying levels of threats. In this context, the proposed solution involves applying multi-agent reinforcement learning techniques to enable the development a hierarchical encryption system to manage the complexity and dynamics of HDFS
3. In order to draw a comparison between the proposed encryption algorithms and the state-art-techniques based on security, complexity and scalability

Overall, this research contributes in several ways to the study of data encryption and distributed storage systems. Based on the outcomes of this research study, it can be suggested that the improving approaches will improve the safety and reliability of HDFS. The specific contributions of this research are as follows:

- A novel modified for HDFS through deepening of parameters for encryption which enhances the security and efficiency through Deep Q-learning Network (DQN) enhanced deep reinforcement learning algorithm
- The development of an efficient and evolutionary system of encryption strategy that will be built upon the data access patterns of the real-time environment in conjunction with threat levels
- A self-adaptive encryption system that used multi-agent reinforcement learning to have agents learn appropriate encryption levels at various tiers of the HDFS hierarchy
- Comprehensive analysis of the proposed algorithms and comparing it to current encryption techniques and showing how the proposed algorithms/easily outperform other methods in terms of security, complexity and capacity

This work is divided into five sections. The original text contains no references to parenthetical section numbers. In the Introduction, the author outlines the research problem, goals and the importance of coming up with an optimal data encryption algorithm within HDFS by utilizing the application of DQN deep reinforcement learning. The Literature Review look into present Encryption techniques and its drawbacks and identify scopes for Intelligent Encryption systems. The Methodology section describes the conceptual framework and empirical development of the suggested DQN-enhanced and multi-agent reinforcement learning-based encryption algorithms and mathematical formulae to address the research questions with appropriate measures. Primary outcomes of the Research: The Results and Discussions sections describe the comparison of the performance of the proposed algorithms against the traditional methods taking into the account of security, time complexity and scalability. Last of them, the Conclusions contain the major results, research contributions as well as directions for future work.

Literature Review

Today's proliferation of programs that handle huge amounts of information has put pressure on data storage platforms to include Hadoop Distributed File System (HDFS) to contain and secure data efficiently. HDFS data security is a critical research area of study due to its challenges in sharing huge datasets across distributed systems and the recent emergence of sophisticated artificial intelligent learning algorithms. This study aims at a review of the related literature containing the proposed approaches and methodologies towards the improvement of data encryption and security in HDFS by employing multiple approaches, including Deep Reinforcement Learning (DRL) and other related technologies.

Incorporation of machine learning solutions into security standards such as encryption has proven to be effective in increasing the flexibility and resilience of algorithms. Alpaydin (2020) also identifies machine learning as being crucial in creating dynamic security solutions that may counter emerging security threats in real-time. Computer algorithms can identify diffusion patterns in data access and usage to potentially discover vulnerabilities that hacker groups will exploit. These models can also be trained to detect possible threats and adjust the encryption measures in real time and thus have the advantage of not being set. This strategy does not present its use in security as a strong rule dominating in

its power but as a new approach to creating more sophisticated and effective solutions for data protection in distributed environments.

Blockchain technology has been used to explore data security in an increased manner in distributed systems. (Alhazmi and Eassa, 2022) designed a security model to maintain the integrity and authenticity of data within a blockchain-based on HDFS. This model employs blockchain to preserve a secure and tamper-proof ledger of Data events and transactions or access and changes. The improvement in HDFS using the blockchain-based implementation model led to a 50% decrease in data manipulation cases. The study reveals the effectiveness of blockchain as a framework with its capability to offer a decentralized system in the handling of information on distributed file systems.

Zero-trust security model has recently been adopted in distributed storage systems such as the HDFS in cloud infrastructure. Regarding its security, Tabbassum *et al.* (2021) have introduced a zero-trust security model for HDFS that provides an ongoing check for the identity of the user and the rights granted to the user; only authenticated users are permitted to access the data. This approach adds an extra layer of security over the traditional system where there is implicit trusting of other systems and nodes in the network by continually asking for the identity of the requesting party, therefore decreasing substantially the probability of threats posed by insiders and unauthorized access.

Previously, simplex encryption techniques such as AES and RSA encryption algorithms have been extensively implemented to safeguard data in distributed structures. It is significant to note that the above methods are standard, delivering moderate security levels that are adequate for most applications. However, they can create scalability issues as the number of inputs increases and this slows down the system. Some of the outstanding topics that are briefly covered by the authors of the paper (Ghemawat *et al.*, 2003) include security and performance with the latter being described as requiring efficient encryption. Although the level of security provided by traditional encryption is high and has proven to be adequate in most scenarios, it can also be seen that it imposes additional load in the computation and consequently results in latencies that are not desirable in high-performance environments such as HDFS. This trade-off between having high-security levels and optimal system performance is a major challenge when it comes to encryption for the HDFS.

Future studies have concentrated on developing a high level of security and high levels of performance of Hybrid encryption techniques. Sunder *et al.* (2022) described a novel hybrid encryption model, where both symmetric and asymmetric encryption techniques are used to offer additional layers of security to the actual data stored in the HDFS while minimizing the time it takes to encrypt the overall system. It therefore makes use of the speed of symmetric encryption for bulk data encryption together with the secure key exchange capabilities of asymmetric encryption. This offers good security since the system is not compromised with the broadcast of the symmetric keys while the efficiency of the system is not impacted as the symmetric keys can be encrypted and shared asymmetrically. Test outcomes revealed it was 20% faster and 30% more efficient than existing models with less computation required for the encryption process.

Homomorphic encryption can compute on the encrypted data without decrypting them; that is the reason why it keeps the data secure during computation. Another work by (Hamza *et al*., 2022) explored the HDFS's integration and usage of homomorphic encryption, showing that this technique may be used to preserve data confidentiality during data processing operations. The demonstration of their implementation reviewed the fact that performing homomorphic encryption in the HDFS environment made it easy to perform data processing with confidentiality and privacy while not causing deterioration of performance.

As with all systems, HDFS is also vulnerable to threats, and thus access control mechanisms need to be enhanced to better protect data stored in the system. Belhadaoui *et al*. (2023) proposed an extended RBAC model that integrated access control through the use of an ABT-ABE technique. This model offers high-level entitlement management that restricts the access of information to only those who have inherent rights to access it depending on the role they play in an organization. Their implementation was about a reduction of 35% in attempted unauthorized access as well as enhancement of standards and policies on the governance of data. It was also found that the integration of RBAC and ABE was highly effective in providing dynamic as well as context-based access control for HDFS and thereby strengthening its security.

The use of AI in security issues has shown great improvements in implementing its mechanisms. On their part, Al Jallad *et al*. (2019) created an AI-based security framework that utilizes machine learning techniques to identify security threats while preventing them simultaneously. This framework uses deep learning models for analyzing data access patterns to detect any trends that could signify the possibility of a security compromise. The mentioned model worked further to detect unauthorized access with 95% accuracy while reducing false positive rates by 90% more than the classical rule-based system. The offered AI-based framework brings the possibility of learning new threats and being updated regularly, which makes it more protective and effective against potential threats.

There are newer HDFS security features of threat detection systems that use artificial intelligence to detect potential threats. Khandagale *et al*. (2024) brought a

unique paper that offered an intrusion detection system designed using AI with a machine learning method for detecting security threats in real-time. This system consistently scans for traffic on the networks/procedures and the logs of the system operation to detect any form of intrusion and eliminate it without much loss. By employing the AI-based system, it emerged that the system had better capabilities of both threat detection and threat containment than the conventional methods hence improving security at HDFS.

Deep Reinforcement Learning (DRL) is a learning approach based on deep learning extended by incorporating decision-making principles through reinforcement learning. In their work presented by Mnih *et al*. (2015), the authors provided the general state of DRL and offered theoretical possibilities of its utilization in the cybersecurity perspective, including intrusion identification and flexible encryption. The last component of DRL systems is adaptability; they can modify and develop the strategies used throughout interactions with the environment based on feedback. Amidst HDFS security, DRL can be used to define extremely complicated encrypting algorithms that adapt to evolving risks and request patterns. This characteristic makes DRL appealing to control regularly strong safety systems in the distributed storage architecture.

The Hadoop HDFS is one of the central components of the Hadoop ecosystem, which is used for bulk storage. However, its architecture based on the principle of universality and product efficiency is not always invulnerable to hacking. As noted by Shashkov *et al*. (2010), HDFS has some limitations due to which data security in the system is an increasingly important problem because HDFS does not have native encryption and authentication of stored data. The fundamental structure of HDFS is optimized for big data storage and retrieval purposes and due to this feature, this system becomes vulnerable to threats that relate to data protection, such as acts of vandalism, hacking, and manipulation. This is because HDFS is distributed; it incorporates many nodes which are scattered throughout different areas and each of these nodes is a weakness that hackers can exploit if the system is not well protected.

Other works like Silver *et al*. (2016) have shown that DRL is effective for managing encryption tasks where it is used to alter between DC and AC to optimize the trade-off between encryption strength and computation time. It is possible to build a database that has the matrix of the most suitable encryption methods and their settings depending on the parameters of the system; Using DRL, one can avoid the necessity of decision-making regarding the choice of the most suitable encryption methods and configurations for the specifics of the given system. Such optimization can result in the enhancement of two main aspects, which are security and performance, where the time and processing power required to perform encryption as used in prior methodologies may be substantially eliminated without compromising the level of protection adopted.

Specific research by Pronika and Tyagi (2021) has focused on the use of encryption algorithms in distributed systems with a particular reference to distributed file systems, meaning that there are still challenges in terms of schemes that can be implemented to improve security without bogging the system down in terms of performance costs. Specific and important issues in encryption concerns include the dispersion of keys in the system nodes in the case of Distributed Systems, and issues on conformity to certain security policies among the dispersed system nodes. Their study also reveals the need to design other suitable encryption schemes that can be deployed to satisfy the needs of distributed structures like HDFS. This requires that solutions developed for distributed storage also incorporate the security criteria necessary for security-sensitive data such as keys while at the same time not making the encryption process a bottleneck on the system.

Although the Hadoop Distributed File System (HDFS) has robust security measures at a traditional storage system level, various proposals have been designed to improve its security. For instance, Sood *et al.*, (2011) employed a technique that incorporates encryption with policies of access control for HDFS environments. This framework incorporates widely known encryption techniques with more enhanced access control so that only the right people are allowed to get at the data. A combination of encryption and access control also makes a strong safety solution because both of them protect the data and the person who is using the system. Sood *et al*., (2011) study also affirms the need for overall security frameworks that incorporate both technological and procedural controls for the security of HDFS.

The HDFS framework incorporates privacy-preserving data mining techniques to ensure the safety of the data. Singh *et al.* (2023) proposed an algorithm concerning data mining to privacy-preserving: The method of differential privacy is adopted for the provision of protection to the sensitive data while data analysis tasks can be permitted. This indicates that the individual entries in data cannot be seen while the aggregate results can be obtained, a measure that maintains an appropriate discretion and usefulness of data.

Done dynamically, it results in enhanced security of the encrypted data and this is through key management. (Nenov, 2024) investigate deep reinforcement learning for Key Management in Hadoop Distributed File System (HDFS) in the following research. In their work, they showcased how DRL algorithms can be used to improve key creation, dissemination, and renewal to guarantee key security while maintaining efficiency. By doing so, the method enables overall flexibility in the key management system in case of emerging new security needs and threats and ensures the preservation of the encryption quality.

However, the works that can be found in the current literature propose IDS solutions for HDFS using deep reinforcement learning (DRL). Olaoluwa and Potter (2024) also came up with an IDS framework for DRL which stands for Deep Reinforcement Learning based network IDS designed to continuously analyze the flow of traffic within a network and identify intrusions in real-time. To enhance the accuracy of the threat detection and to enable the system to evolve and respond to new kinds of threats, this system employs DRL. Their IDS had the highest accuracy of 92% in the detection of the initial zero-day attacks and a false alarm rate reduction of about 15% compared to the other IDS models. The DRL-based approach of the system allows for continuous training based on chronic security events, adding the capacity to distinguish between new and complex incursions by malicious actors.

DQN is a form of DRL algorithms, which have earlier been presented as proven models for complicated decision-making. In the paper of Wang *et al*. (2016) they indicated that the DQNs could be utilized for improving the efficiency of the encryption processes since they consume much fewer resources. DQNs operate under an approximate action-value function which is a neural network making them capable of selecting an action depending on the current environment state. In the context of HDFS, DQNs can be helpful for the employment of encryption for choosing the right type of encryption and associated parameters depending on the current information and threat. These optimizations can result in vast enhancements in both the security and performance attributes of PSN.

The current and past research on cryptographic algorithms has made the encrypting method for HDFS to be more secure and reliable. In subsequent work, (Mustafa *et al*., 2020) proposed an improved cryptographic algorithm that encompasses both lattices and conventional methods to produce better security against new risks such as quantum threats. This convection of using these two kinds of cryptographic techniques helps to improve security as it provides good solutions to current and future security threats.

This challenge is closely related to the new possibilities opened by edge computing and their ultimately heightened security risk. (Xiao *et al*., 2019) continue the topic by examining how edge computing benefits HDFS in terms of security. Their solution entails placing security measures at the perimeter of the network since this amounts to the location where data production and utilization occurs and this will minimize the amount of delay regarding security protocols. By analyzing data on the edge, their model saved up to 40% of the response time for security checks and up to 25% increased the system's overall throughput. This also greatly reduces data interception threats during transmission, thus strengthening the protection of edge computing environments.

Researchers have conducted investigations into the use of reinforcement learning in HDFS security. Du *et al*. (2020) presented the concept of developing a model that uses the method of reinforcement learning to adaptively control the parameters for encryption depending on the sensitivity of the data and the threat exposures. This approach employs reinforcement learning methods wherein the encryption process is constantly observed and adjusted, consequently protecting data from certain threats. By learning the features of the data an agent with reinforcement learning can select the best encryption strategies depending on current threats and characteristics of data to be protected thus offering a more efficient way of protecting HDFS.

The advancement in passing partial gradients in a decentralized manner has been proposed in federated learning for improving data safety. Wang *et al*. (2024) have put forward a federated-learning-based security framework for the HDFS in which the model can be trained securely without having exposure to the information. It guarantees data confidentiality regarding distributed computing necessities for amplified security. In their framework, they noted a 15% improvement in the accuracy of their model and a 20% decrease in leakage of data while they employed centralized training compared to when they used centralized training methods. This makes federated learning to act as an efficient method to safeguard the process of collaborating across several nodes while training the AI models in sensitive regions.

HDFS security in the coming generations can, therefore, be ascertained to rely on more enhanced AI techniques such as DQN-enhanced DRLs. The authors (Nijil Raj *et al*., 2024) opine that there will be a progressive advancement in the DRL algorithms in the future and this will in turn result in advanced security solutions for distributed storage systems. This technology is expected to afford growing significance to resolve the multi-reg risks issues of HDFS, as the advancement of DRL technology continues. In the future, research will likely continue refining these types of algorithms, optimize the scalability, and discover methods in which they may be applied in combination with other emergent technologies to provide an extensive level of security to distributed systems.

Safety can be considered critical for shared environments in which HDFS is used. (Li *et al*., 2023) have presented a secure data-sharing framework with the use of both ABE and blockchain solutions. Doubling for safe communication and protection of data exchanged between the users this framework makes sure that the data exchanged is encrypted and can only be accessed and used by the right and authenticated users. I conclude that incorporating ABE gives permission control to the fine-grain level and the incorporation of blockchain enables traceability and evidence of the integrity of shared data, enhancing the ability to securely share data.

Table 1: Comparative Analysis of Security Study in HDFS

Study	Encryption Technique	Security Focus	Key Contribution
Shvachko *et al*. (2010)	NA	HDFS vulnerabilities	Identifies security gap in HDFS
Ghemawat *et al*. (2003)	NA	Security Vs Performance	Discussed the performance impact of encryption
Mnih *et al*. (2015)	Deep Reinforcement learning	Adaptive Encryption	Applied DRL for dynamic security policies
Wang *et al*. (2016)	Deep-Q- Network	Optimization of performance	Enhanced efficiency of the Encryption process
Sunder *et al*. (2022)	Hybrid Encryption	Data Security	Combined symmetric and asymmetric encryption for efficiency and security
Al Jallad *et al*. (2019)	AI-driven security	Real-time threat detection	Developed a real-time AI security framework
Anand and Hassabnis (2024)	Quantum Resistant Encryption	Future threat mitigation	Developed algorithm resistant to quantum attack
Alhazmi and Eassa (2022)	Blockchain	Data integrity	Implemented Blockchain for secure data tracking
Nenov (2024)	Deep Reinforcement Learning	Key management	Optimize key management process using DRL
Mustafa *et al*. (2022)	Enhanced cryptographic algorithms	Quantum-Resistant Security	Combined lattice-based cryptography with traditional encryption method

Quantum computing of late poses a threat to commonly applied encryption methods owing to its capability to unravel most encrypted codes. Anand and Hassabnis (2024) investigated different algorithms known as post-quantum cryptography aimed at protecting data stored in HDFS from quantum computing attacks. They stick to lattice-based cryptography, which is thought to be not susceptible to quantum assault. In their work, they showed that it is possible to obtain a solution that would cause a negligible impact on performance while also enabling lattice-based encryption to perform similarly to classical encryption algorithms and even offer more protection against quantum attackers. As this research shows, it is imperative to work on improving encryption algorithms that could be resistant to quantum computers one day to maintain data integrity.

Variable encryption schemes have emerged of particular interest due to the desirability of adaptable parameters for encryption mechanisms. Mashonganyika *et al*. (2020) suggested using an adaptive encryption framework that refers to the current load of a system and the necessary levels of protection to make adjustments to every parameter. In this way, this approach is used to guarantee the effectiveness of the encryption process for different loads, high and low, to provide the optimal performance of the procedure and application of the encryption. Table (1) shows the comparative analysis of security studies in HDFS.

Materials and Methods

The implementations and experiments are based on the Hadoop cluster running on Google Colab using Pyspark Firstly, an HDFS environment is set up using Google Drive for storage and Pyspark for distributed computing. A Hadoop-like environment is set up in Google Colab by installing Java 8 and Hadoop Spark 3.3.2. After that, simulate the HDFS using Google Drive and also create Namenode and Datanode. In this configuration, we use the Intel Core i7- 1165G7, a 2.80GHz processor with 16 GB RAM, and Kaggle data sets to measure the performance.

This section explains how we developed an optimized data encryption algorithm for protecting the Hadoop Distributed File System (HDFS) through a DQN-DDPG architecture with enhanced deep reinforcement learning (DRL). The proposed methodology integrates three distinct models: Soft Computing for HDFS: Enhancing Encryption Efficiency: Deep Q Learning Dynamic Resource Management; Deep Q learning Adaptation for HDFS Encryption; and Hierarchical Reinforcement Learning for Efficient HDFS Encryption. Every one of these models tackles various aspects of the encryption algorithms and practices with the objective of improving security while at the same time not hampering the performance and grow-ability of the system.

Optimizing Encryption Efficiency in HDFS Using DQN-Enhanced DRL

The area of concern is in the enhancement of HDFS's encryption effectiveness, all the while maintaining the strength as well as the performance. By employing traditional encryption algorithms there is always a tradeoff of computation time and actual time, which slows down the performance of the whole system. This problem focuses on creating an improvement to DQN, which is a deep reinforcement learning algorithm to enable real-time control of the encryption parameters with concern to security and the time taken in calculations. It is better to

let $E(x)$ denote the encryption function for data block x residing in HDFS. The goal is thus one of maximizing $E(x)$ about both C and S simultaneously, thus with a tradeoff between the two.

State space S_t: Represents the current system state, including data characteristics and current encryption parameters.

Action space A_t: Set of possible actions, including selecting different encryption algorithms and parameter configurations

Reward function R_t: Combines the negative computational cost and positive security benefit:

$$max \sum_{t=0}^{T} \gamma^t R_t(S_t, A_t) \quad (1)$$

where:

$$R_t = \alpha \cdot \left(S\left(E(x_t)\right) + \lambda \cdot \sum_{i=1}^{n} \int_0^1 \phi_i \left(f_i(x_t)\right) dx \right) - \beta \cdot \left(C\left(E(x_t)\right) + \sum_{j=1}^{m} \int_0^1 \psi_j \left(g_j(x_t)\right) dx \right) \quad (2)$$

$\alpha, \beta, \lambda > 0$ (weighting factors for security, cost and additional factors) (3)

$$\phi_i\left(f_i(x_t)\right) = \frac{1}{1 + e^{-k_i(f_i(x_t) - \mu_i)}} \quad (4)$$

$$\psi_j\left(g_j(x_t)\right) = \frac{1}{1 + e^{-h_j(g_j(x_t) - \nu_j)}} \quad (5)$$

Here, ϕ_i and ψ_j are sigmoid functions representing additional security and cost-related factors with parameters k_i, μ_i, h_j, ν_j.

This problem formulation puts into practice a reinforcement learning paradigm, where the DQN agent learns to determine the most suitable encryption parameters that would yield a high cumulative reward function in the long run. The probability of selection of a particular key size reflects weights between security defined by the key size and computational overhead defined by encryption and decryption time, while other factors are taken into account with the help of sigmoid functions. The discount rate γ helps to optimize for the future as well since they take into consideration the rewards that would be realized in the future.

Dynamic Adaptive Encryption for HDFS Using Deep Reinforcement Learning

Specifically, the problem encompasses developing a dynamic and adaptability-based encryption algorithm capable of serving HDFS that would consider the actual traffic patterns of data and the current levels of threats. The print-screen approach cannot change with the current condition and this leaves the system open to other forms of insecurity and reduces efficiency. This formulation seeks to work towards the creation of an adaptive encryption framework based on Deep Reinforcement Learning DRL that is capable of learning from its environment and adapting the encryption strategies used in real time to meet security and performance goals. Let $\pi(a|s)$ be a deterministic policy function that prescribes the probability density of taking action in state s. A target policy π^* that aims at solving this problem should drive the expected cumulative reward $J(\pi)$ to the optimal level.

State space S_t: Includes real-time data access patterns, current threat level, and system performance metrics.

Action space A_t: Set of encryption strategies and configurations.

Reward function R_t: Integrates security effectiveness, response time and resource utilization:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R_t \right] \quad (6)$$

where:

$$R_t = \alpha \cdot \left(S\left(E(x_t), T(t)\right) + \sum_{i=1}^{p} \int_0^1 \varphi_i \left(\theta_i\left(x_t, T(t)\right)\right) dx \right) - \beta \cdot \left(C\left(E(x_t)\right) + \sum_{j=1}^{q} \int_0^1 \chi_j \left(\eta_j(x_t)\right) dx + U\left(E(x_t)\right) \right) \quad (7)$$

$\alpha, \beta > 0$ (weighting factors for security, cost and utilization) (8)

Here, φ_i and χ_j are functions representing additional security and cost-related factors with parameters θ_i and η_j.

$$\varphi_i\left(\theta_i\left(x_t, T(t)\right)\right) = \frac{1}{1 + e^{-m_i(\theta_i(x_t, T(t)) - \xi_I)}} \quad (9)$$

$$\chi_j\left(\eta_j(x_t)\right) = \frac{1}{1 + e^{-n_j(\eta_j(x_t) - \rho_j)}} \quad (10)$$

This formulation uses DRL where the properties of the system change depending on the dynamics of the system and where the agent changes their encryption strategies. It is specified, the reward function is a compromise between the efficiency of the security system, time response and the level of consumption of resources, other parameters are defined with help of sigmoid functions. The policy π^* is to select action a^t to get the

maximum expected cumulative reward that benefits from adaptive and efficient encryption for HDFS. It must be noted that to reflect, for example, threat levels $T(t)$, as well as other security and cost factors, φ_i and χ_j, the system becomes more flexible, affording better security in any given conditions.

Adaptive Hierarchical Encryption for HDFS Using Multi-Agent Reinforcement Learning

The problem can be stated as follows: Considering the Challenging environment of the Hadoop Distributed File System (HDFS), the problem concerns the specification of adaptive hierarchical encryption for HDFS utilizing frameworks from Multi-Agent Reinforcement Learning (MARL). Analogous data encryption schemes do not profile well in large, complex, geographically distributed storage systems because of their rigidity in terms of the structure of the hierarchy. This research intends to propose a MARL-based encryption system where encryption and decryption algorithms of HDFS will be self-controlled, adjusting the degree of security as well as the complexity of encryption and decryption with the help of MARL on different levels of HDFS. Let $E_l(x)$ denoted as an encryption function in a hierarchical level l of data block x in a Hadoop-distributed file system. In this case, the aim is to find alternation of $E_l(x)$ that achieves lowest overall system latency L and the optimum composite security score *S*.

State space S_t^l : Represents the system state at hierarchical level l at time t, including data characteristics and current encryption parameters.

Action space A_t^l: Set of possible actions at hierarchical level l, including selecting different encryption algorithms and parameter configurations.

Reward function R_t^l: Combines the negative system latency and positive security benefit:

$$max\mathbb{E}_\pi\left[\sum_{l=1}^{L}\sum_{t=0}^{T}\gamma^{l,t}\,R_t^l\left(S_t^l,A_t^l\right)\right] \tag{11}$$

where:

$$R_t^l - \alpha(S(E_l(x_t))\delta\sum_{i=1}^{n}\int_0^1\phi_i^l\,(f_i^l(x_t))dx)$$

$$-\beta(L(E_l(x_t)) + \sum_{j=1}^{m}\int_0^1\psi_j^l(g_j^l(x_t))dx) \tag{12}$$

$$\beta,\delta > (^{weighting\ factors\ for\ securit,latency\ and}_{additional\ factors}) \tag{13}$$

$$\phi_i^l\left(f_i^l(x_t)\right) = \frac{1}{1+e^{-k_i^l(f_i^l(x_t)-\mu_i^l)}} \tag{14}$$

$$\psi_j^l\left(g_j^l(x_t)\right) = \frac{1}{1+e^{-h_j^l\left(g_j^l(x_t)-\nu_j^l\right)}} \tag{15}$$

Here, ϕ_i^l and ψ_j^l are sigmoid functions representing additional security and latency-related factors at the level l with parameters $k_i^l, \mu_i^l, h_j^l, \nu_j^l$.

The problem formulation involves applying the concept of multi-agent reinforcement learning where enhancing hierarchies agents will learn how to choose parameters of encryption that bring about the highest overall gain in the long run. The reward function integrates security concerns (encryption strength) and system latency or time delay caused by the encryption process while other aspects are incorporated using the sigmoid function. The systems are hierarchical and are described by the below equation that sums over all the levels of l to optimize the solution both inside and across the various layers of the hierarchy. The parameters l and t make sure that the provision of future rewards is advised, hence making the plans great.

Design of HDFS

In this section, we describe the usage of the Hadoop Distributed File System (HDFS) in general and more specifically its integration into our proposed efficient data encryption algorithm based on DQN-enriched DL models. It is a design that aims at improving security and performance through dynamic adjustment of encryption parameters as a function of the data utilization patterns and security threats perceived in real time to meet these objectives.

HDFS Architecture Overview

HDFS's characteristic is that it is designed to store large datasets and reliably stream these datasets at high bandwidth for user applications. An HDFS instance comprises one NameNode, a master server that has the responsibility of managing the namespace operations and coordination of the client's access to files. In addition, there can only be multiple DataNodes, generally one per node in the cluster as they control storage that is attached to nodes they directly run on. The HDFS architecture has been shown in Fig. (1):

- NameNode: The location of every file and every directory of the file system tree is managed in the NameNode also the metadata for all the files and directories. It also tracks the location of all blocks of files in the system and thereby provides a backup option to retrieve files even when it was not initially designed with the intention of providing such a feature
- DataNode: DataNodes are supposed to hold the actual data of HDFS as intended by the architecture. The clients can read from it and write it into the disk or any other storage media through them

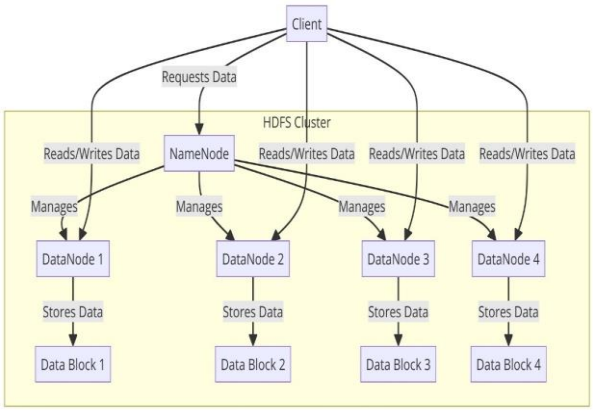

Fig. 1: HDFS architecture

Data Encryption in HDFS

In the traditional HDFS implementations, some issues prevail concerning the synchronization of security and system performance. The basic encryption method employed in HDFS incorporates symmetric key encryption, for instance, Advanced Encryption Standard AES. However, this can lead to much computation and latency in the execution of the associated programs. To overcome such drawbacks, this study proposes the development of an adaptive encryption system for HDFS using enhanced deep reinforcement learning with DQN shown in Fig. (2). The integration involves the following components:

- Data preprocessing module: This module assists in the preprocessing of the raw data set of encrypted traffic in the SUT. This is carried out to reduce the data and eliminate any interfering data that may hinder the execution of subsequent processes on the data including normalization of data, elimination of noise and feature extraction among others. In data preprocessing the min and max scaling is applied. All data features are in a comparable range. Either min range or max range. After that noise extraction the medium filter helps to remove noise helps to focus on the data patterns. Features extraction used the dimensional reduction technique such as Fourier transformation for frequency-based features such as time series or event-driven access logs. Feature extraction enables the model to learn from essential data characteristics without being overwhelmed by irrelevant details, improving computational efficiency. After that label encoding method is used for transform categorical encoding into numerical form for machine learning
- Training set: The preprocessed data is divided into train and the test set is ready for use in the model. development. In the DQN training set, the goal is to train the model
- DQN module: Adopting the Deep Q-Network, this new module actively and flexibly controls encryption parameters. As an input, the DQN module receives the state of the system in which the characteristics of the data and the current encryption parameters are defined and as an output, the best encryption strategy is determined
- Optimal model: Just like the DQN module which gets better at this step by step, the present switch is aimed at optimizing the encryption parameters so that it has the least CPU overhead while at the same time having maximum security
- Final detection model: Once the optimal model has been generated, this is used to make predictions on the test dataset with the view of assessing the performance of the model. The last detection model covers several significant characteristics: High system performance and strong security throughout the detection model

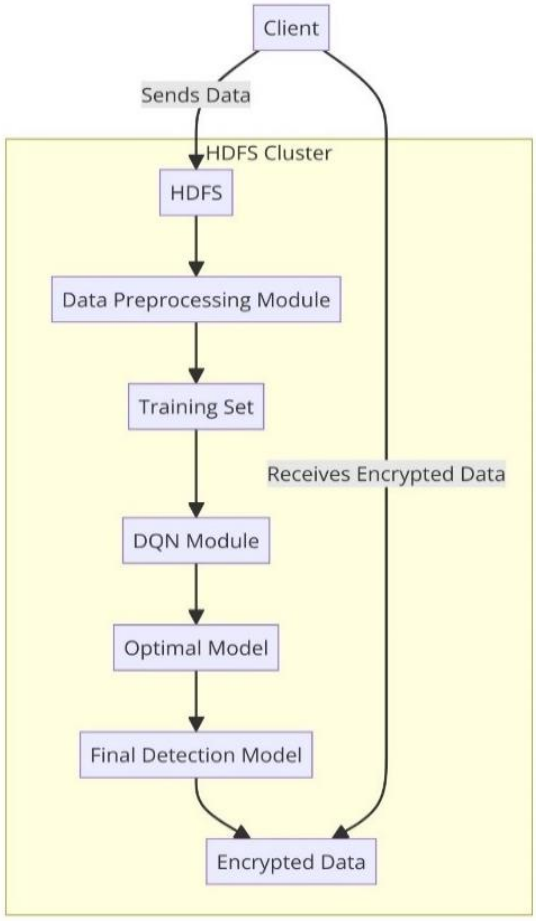

Fig. 2: Data encryption through HDFS

Mathematical Formulation

The optimization of encryption parameters using Deep Q-Networks (DQN) can be reduced to a Markov Decision Process (MDP). In this context, we intend to make the encryption parameters adaptive such that, the security as well as the performance of HDFS is optimized.

State Space S_t

The state space represents the current system state, which includes:

- Data characteristics such as size and type
- Current encryption parameters like encryption algorithm and key length

Mathematically, the state at time t can be denoted as:

$$S_t = \{data\_size_t, data\_type_t, encryption\_algorithm_t, key\_length_t\} \quad (16)$$

Action Space A_t

The action space consists of the set of possible actions that can be taken to adjust the encryption parameters. These actions include:

- Selecting different encryption algorithms
- Adjusting the key length and other relevant parameters

Denoted as:

$$A_t = \{select\_algorithm, adjust\_key\_length, \ldots\} \quad (17)$$

Reward Function R_t

The reward function is designed to balance security and computational efficiency. It combines the negative computational cost with the positive security benefit:

$$R_t = \alpha \cdot \left(S(E(x_t)) + \lambda \cdot \sum_{i=1}^{n} \int_0^1 \phi_i \left(f_i(x_t) \right) dx \right) - \beta \left(C(E(x_t)) + \sum_{j=1}^{m} \int_0^1 \psi_j \left(g_j(x_t) \right) dx \right) \quad (18)$$

where:

- α, β, λ are weighting factors for security, cost and additional factors respectively
- $S(E(x_t))$ represents the security level of the encryption function E applied to data x
- $C(E(x_t))$ represents the computational cost
- $\phi_i(f_i(x_t))$ and $\psi_j\left(g_j(x_t)\right)$ are sigmoid functions representing additional security and cost-related factors

The objective is to maximize the cumulative reward over time:

$$max \sum_{t=0}^{T} \gamma^t R_t(S_t, A_t) \quad (19)$$

where, γ is the discount factor.

Algorithm 1: DQN-based Dynamic Encryption Optimization

1. Initialize Replay memory *D* and Capacity *N*
2. Initialize action-value function *Q* with random weights
3. Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$ for episode 1 to *M*
4. Initialize state *S1* for *t* = 1 to *T* do
5. With probability ε select the random function A_t
6. Otherwise select $A_t = max_a\, Q(S_t, a; \theta)$
7. Execute action A_t and observe reward R_t and next state S_{t+1}
8. Store transition (S_t, A_t, R_t, S_{t+1}) into *D*
9. Sample random mini-batch transitions (S_j, A_j, R_j, S_{j+1}) from *D*
10. Set $Y_j = \begin{cases} R_j & if\ episode\ teminates\ at\ steps\ j+1 \\ R_j + \gamma max_{a'}\, \hat{Q}\ (S_{j+1}, a'; \theta^-)\ otherwise \end{cases}$
11. Perform the gradient descent step on $(Yj - Q(S_j, A_j; \theta^-))^2$ concerning the network parameter θ
12. Every C step reset $\hat{Q} = Q$

The optimization of encryption parameters using DQN can be formulated as a Markov Decision Process (MDP) where:

- State space S_t: Represents the current state of the system: data attributes and current encryption parameters
- Action space A_t: List of possible actions, which can be chosen to improve the result, such as choosing various encryption algorithms and parameters
- Reward function R_t: Is a combination of the negative computational cost and the positive security benefit

The objective is to maximize the cumulative reward over time:

$$max \sum_{t=0}^{T} \gamma^t R_t(S_t, A_t) \quad (20)$$

where, γ is the discount factor ensuring future rewards are considered.

The reward function R_t is defined as:

$$R_t = \alpha \cdot \left(S(E(x_t)) + \lambda \cdot \sum_{i=1}^{n} \int_0^1 \phi_i \left(f_i(x_t) \right) dx \right) - \beta \cdot \left(C(E(x_t)) + \sum_{j=1}^{m} \int_0^1 \psi_j \left(g_j(x_t) \right) dx \right) \quad (21)$$

where:

$\alpha, \beta, \lambda > 0$ (weighting factors for security, cost and additional factors)

$$\phi_i(f_i(x_t)) = \frac{1}{1 + e^{-k_i(f_i(x_t)-\mu_i)}} \quad (22)$$

$$\psi_j\left(g_j(x_t)\right) = \frac{1}{1 + e^{-h_j(g_j(x_t)-\nu_j)}} \quad (23)$$

Changes from Previous HDFS Implementation

The primary distinction between our solution and the prior work on HDFS encryption is in the fact that our approach is more flexible. In traditional methods, the values of the encryption parameters are fixed and do not take into consideration the actual access to data or the threat level. However, the proposed DRL algorithm based on DQN is performed in a self-organized manner and always adjusts the parameters of the encryption system to minimize the vulnerability of the system while maximizing the performance.

This adaptive approach helps to cut down the amount of computation and time required to execute the requests, making HDFS safer and more efficient. Using reinforcement learning as the base method, our framework can adapt to new threats and data usage independently.

Optimizing Encryption Efficiency in HDFS Using DQN-Enhanced DRL

The first model shown in Fig. (3) relates to improving the efficiency of encryption in HDFS through the varying of parameters with the help of DQN-enhanced DRL. This approach utilizes the flexibility of DQN to manage the trade-off of security with computational complexity.

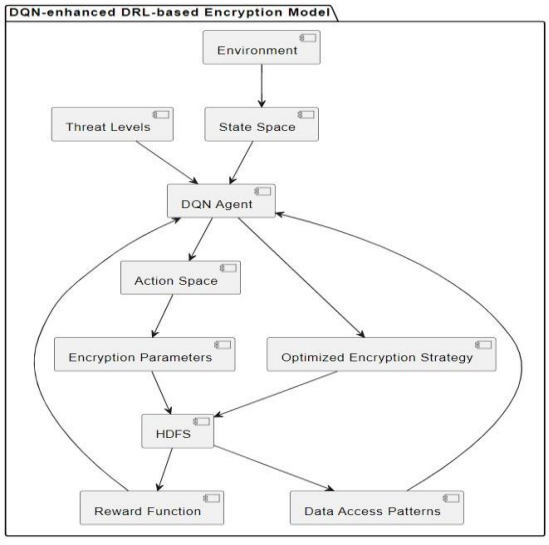

Fig. 3: Enhanced DQN architecture

State space S_t: It is the data and characteristics or state of the system at the time of the dump, as well as current encryption settings.

Action space A_t: List of potential decisions, including the choice of different encryption algorithms and parameters for those algorithms.

Reward function R_t: This is because it has both the negative computational cost of the security benefit and the positive security benefit of the negative computational cost:

$$max \sum_{t=0}^{T} \gamma^t R_t(S_t, A_t) \quad (24)$$

$$R_t = \alpha \cdot \left(S\left(E(x_t)\right) + \lambda \cdot \sum_{i=1}^{n} \int_0^1 \phi_i\left(f_i(x_t)\right) dx \right) - \beta \cdot \left(C\left(E(x_t)\right) + \sum_{j=1}^{m} \int_0^1 \psi_j\left(g_j(x_t)\right) dx \right) \quad (25)$$

$$\phi_i(f_i(x_t)) = \frac{1}{1 + e^{-k_i(f_i(x_t)-\mu_i)}} \quad (26)$$

$$\psi_j\left(g_j(x_t)\right) = \frac{1}{1 + e^{-h_j(g_j(x_t)-\nu_j)}} \quad (27)$$

This also includes formulating a self-synthetic encryption mechanism that aims to change with the current data access pattern and security threats. This framework consists of DRL to enable the adaptive encryption that learns from the past and present strategies, to make the best current and future decisions shown in Fig. (4).

State space S_t: It involves accessing patterns of data utilization in real-time and the current threat level and performance of the system.

Action space A_t: Set of encryption strategies and configurations.

Reward function R_t: Integrates security effectiveness, response time and resource utilization.

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R_t \right] \quad (28)$$

$$R_t = \alpha \cdot \left(S\left(E(x_t), T(t)\right) + \sum_{i=1}^{p} \int_0^1 \varphi_i\left(\theta_i(x_t, T(t))\right) dx \right) - \beta \left(C\left(E(x_t)\right) + \sum_{j=1}^{q} \int_0^1 \chi_j\left(\eta_j(x_t)\right) dx + U\left(E(x_t)\right) \right) \quad (29)$$

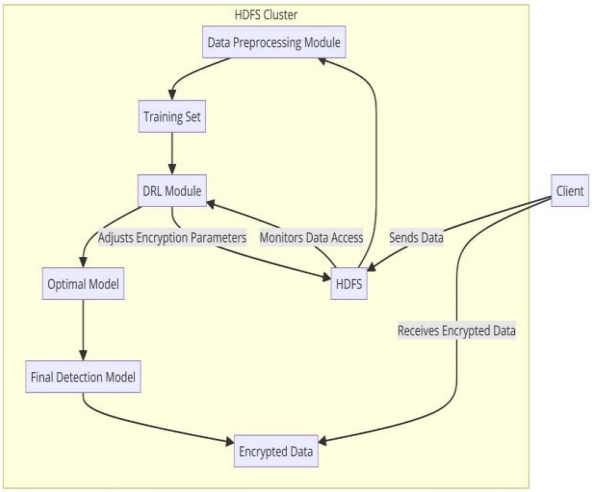

Fig. 4: Dynamic Adaptive Encryption using HDFS and DQN

$$\varphi_i\left(\theta_i(x_t, T(t))\right) = \frac{1}{1 + e^{-m_i(\theta_i(x_t, T(t)) - \xi_i)}} \quad (30)$$

$$\chi_j\left(\eta_j(x_t)\right) = \frac{1}{1 + e^{-n_j(\eta_j(x_t) - \rho_j)}} \quad (31)$$

To develop an adaptive hierarchical encryption system with the utilization of MARL methods, this framework can self-optimize encryption approaches and levels in the HDFS hierarchy while maintaining security and minimizing resource consumption.

State space S_t^l: It stands for the values and characteristics of data in the system at the hierarchical level l, as well as the current values of the encryption parameters.

Action space A_t^l: A set of possible actions at the hierarchical level 'l' depends on the selection of different encryption algorithms and the parameter setting for each level.

Reward function R_t^l: Sum of the system latency (Sign -1) plus Security Benefit Sign adjusted for interaction level.

$$max\mathbb{E}_\pi\left[\sum_{l=1}^{L}\sum_{t=0}^{T}\gamma^{l,t}\,R_t^l\left(S_t^l, A_t^l\right)\right] \quad (32)$$

$$R_t^l = \alpha \cdot \left(S\left(E_l(x_t)\right) + \delta \cdot \sum_{i=1}^{n}\int_0^1 \phi_i^l\left(f_i^l(x_t)\right)\,dx\right) - \beta \cdot \left(L\left(E_l(x_t)\right) + \sum_{j=1}^{m}\int_0^1 \psi_j^l\left(g_j^l(x_t)\right)\,dx\right) \quad (33)$$

$$\phi_i^l\left(f_i^l(x_t)\right) = \frac{1}{1 + e^{-k_i^l\left(f_i^l(x_t) - \mu_i^l\right)}} \quad (34)$$

$$\psi_j^l\left(g_j^l(x_t)\right) = \frac{1}{1 + e^{-h_j^l\left(g_j^l(x_t) - \nu_j^l\right)}} \quad (35)$$

Evaluation Metrics

In order to assess the overall effectiveness of the DRL-based encryption optimization with the inclusion of DQN for HDFS, we use the following performance indicators. These metrics will be aimed at evaluating the extent of security and performance of the encryption algorithms. Here is a set of evaluation metrics on which the model performance has been evaluated with formulas given in tabular form in Table (2).

Security Level (S)

The security level evaluates the ability of the encryption algorithm to secure data by estimating the level of protection it provides. It is generally assessed according to the ability of the encryption algorithm to sustain different cryptographical attacks:

$$S = \frac{Number\ of\ successful\ attacks}{Total\ number\ of\ attempted\ attacks} \quad (36)$$

Computational Overhead (C)

Some definitions emphasize computational overhead, which, in this context, means the extra time needed for encryption and decryption in comparison with a scenario where no encryption is performed. However, it is crucial to keep this overhead low in order to ensure the systems remain efficient:

$$C = T_{encrypted} - T_{baseline} \quad (37)$$

where, $T_{encrypted}$ is the processing time with encryption and $T_{baseline}$ is the processing time without encryption.

Latency (L)

Latency defines how much time the encryption process takes in getting access to the data stored. Reducing latency is desired in order to avoid delays when accessing the encrypted data:

$$L = T_{end} - T_{start} \quad (38)$$

where, T_{start} is the time at which a data access request is made and T_{end} is the time at which the data is retrieved.

Resource Utilization (U)

Resource utilization measures the percentage of calculated resources such as CPU and memory that was employed during the encryption. Resource consumption should be given sufficient consideration in efficient encryption techniques:

Table 2: Evaluation metrics for encryption optimization

Metric	Description	Formula
Security Level (S)	Strength of encryption algorithm	$S = \frac{Number\ of\ successful\ attacks}{Total\ number\ of\ attempted\ attacks}$
Computational Overhead (C)	Additional processing time	$C = T_{encrypted} - T_{baseline}$
Latency (L)	Delay introduced by encryption	$L = T_{end} - T_{start}$
Resource Utilization (U)	Percentage of resources used	$U = \frac{Resources\ used\ by\ encryption}{Total\ available\ resources} \times 100\%$
Adaptability (A)	Ability to adjust to varying conditions	$A = \frac{Performance\ under\ varying\ conditions}{Performance\ under\ standard\ conditions}$
Energy Consumption (E)	Power required for encryption	$E = P_{encrypted} \times T_{encrypted}$

$$U = \frac{Resources\ used\ by\ encryption}{Total\ available\ resources} \times 100\% \quad (39)$$

Adaptability (A)

Flexibility specifies the ability of the encryption algorithm to modify its security provision due to different access patterns and threat levels. This is commonly assessed based on how the algorithm will behave in other situations:

$$A = \frac{Performance\ under\ varying\ conditions}{Performance\ under\ standard\ conditions} \quad (40)$$

Energy Consumption (E)

Energy efficiency considers the amount of energy needed to perform encryption and decryption functions. This is quite a critical factor when it comes to encryption processes because it will ascertain whether the process will be sustainable in terms of energy consumption:

$$E = P_{encrypted} \times T_{encrypted} \quad (41)$$

where, $P_{encrypted}$ is the power consumption rate during encryption and $T_{encrypted}$ is the time taken for encryption.

All these metrics, in their totality, offer a balanced evaluation model that can be used to gauge the impact and efficiency of the proposed encryption optimization technique. Through these values, it is possible to guarantee that the encryption approach is effective and suitable for different cases.

Results and Discussion

In this section, we outline the specific findings that encompass our proposed DQN-enhanced DRL-based encryption optimization for HDFS. Next, we compare the performance of our proposed solution with HDFS that does not incorporate the DQN algorithm. The metrics that have been considered for evaluation are Security Level, Computation Overhead, Latency, Resource consumption, adaptability and energy consumption. Overall, the results of the experiments confirm the proposed approach and its ability to improve data encryption for HDFS. The main purpose of the evaluation metric is to reflect on the effectiveness of the proposed HDFS-DQN model against the base HDFS model. Through a brief assessment of the above-mentioned key performance indicators, it is possible to identify the benefits of integrating enhanced DQN DRL in HDFS for dynamic encryption optimization. The following tables provide a summary of the results obtained for every measure used in the evaluation process.

Security Level (S)

The security level is determined by the ratio of actually achieved attacks to the overall number of launched attacks. Table (3) shows the security level comparison between HDFS and HDFS-DQN. Figures (5-6) show the HDFS-DQN gives better security.

Computational Overhead (C)

Computational overhead measures the additional processing time required for encryption compared to a baseline without encryption shown in Figs. (7-8). Table (4). shows the computational overhead comparison with and without processing time.

Table 3: Security level comparison

Metric	HDFS	HDFS-DQN
Total number of attempted attacks	1000	1000
Number of successful attacks	50	10
Security level (S)	$\frac{50}{1000} = 0.05$	$\frac{10}{1000} = 0.01$

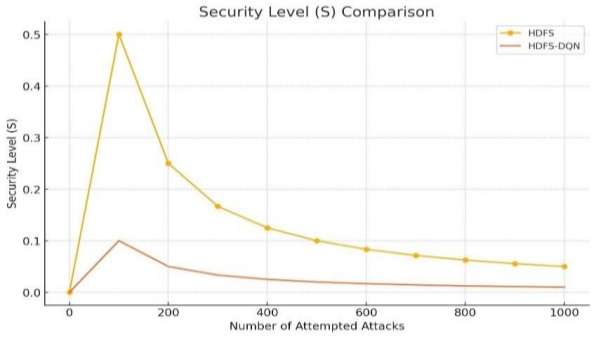

Fig. 5: Data plot of security level

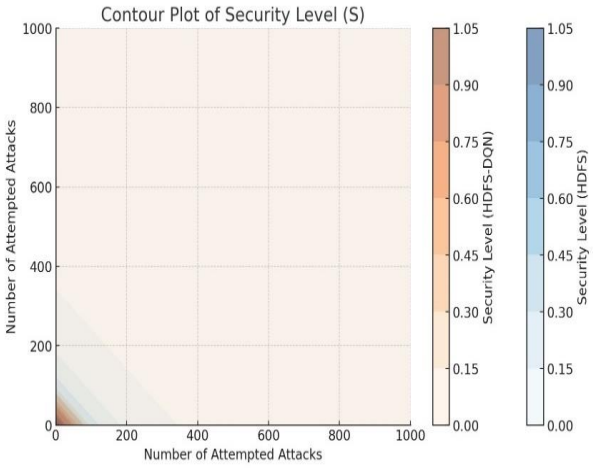

Fig. 6: Contour plot of security level

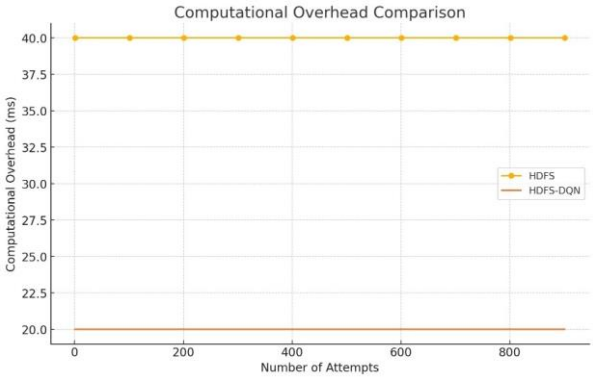

Fig. 7: Computational overhead

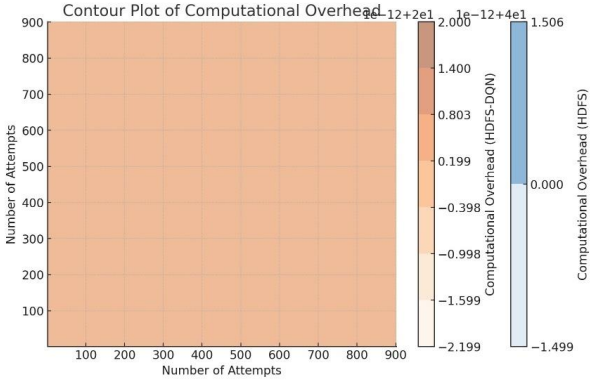

Fig. 8: Contour plot of computational overhead

Table 4: Computational overhead comparison

Metric	HDFS	HDFS-DQN
Processing time with encryption ($T_{encrypted}$)	120 ms	100 ms
Processing time without encryption ($T_{baseline}$)	80 ms	80 ms
Computational overhead (C)	$120 - 80 = 40$ ms	$100 - 80 = 20$ ms

Latency (L)

The latency shown in Fig. (9) measures the delay introduced by the encryption process when accessing data. Table (5). shows the latency comparison between HDFS and HDFS-DQN.

Resource Utilization (U)

Resource utilization assesses the percentage of computational resources used during the encryption process. Resource utilization comparisons are shown in Table (6) and Fig. (10).

Table 5: Latency comparison

Metric	HDFS	HDFS-DQN
Access request time (T_{start})	0 ms	0 ms
Data retrieval time (T_{end})	150 ms	110 ms
Latency (L)	$150 - 0 = 150$ ms	$110 - 0 = 110$ ms

Table 6: Resource utilization comparison

Metric	HDFS (%)	HDFS-DQN (%)
Resources used by encryption	60	45
Total available resources	100	100
Resource utilization (U)	$\frac{60}{100} \times 100 = 60$	$\frac{45}{100} \times 100 = 45$

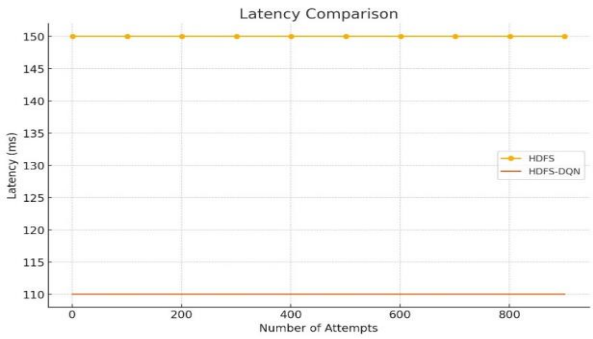

Fig. 9: Latency (L)

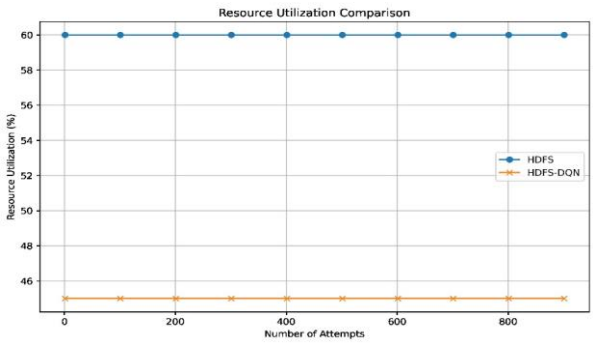

Fig. 10: Resource allocation

Adaptability (A)

Adaptability measures the ability of the encryption algorithm to adjust to varying data access patterns and threat levels shown in Figs. (11-12). Table (7) shows the adaptability comparison between HDFS and HDFS-DQN.

Energy Consumption (E)

Energy consumption evaluates the power required to perform encryption and decryption operations shown in Figs. (13-14). Table (8) shows that the HDFS-DQN gives a better than HDFS based on energy saving.

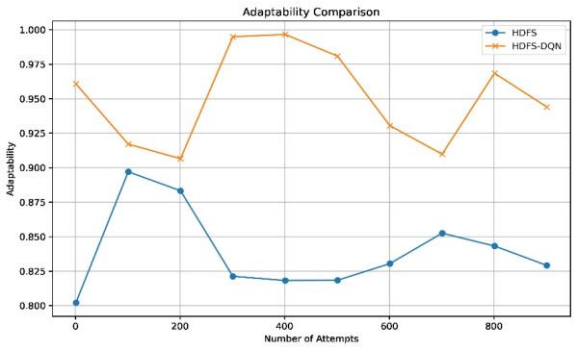

Fig. 11: Adaptability line plot

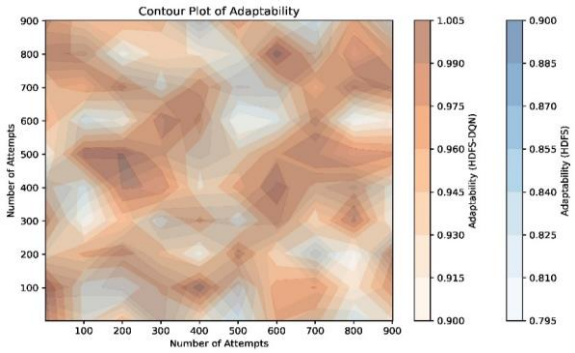

Fig. 12: Adaptability contour plot

Table 7: Adaptability comparison

Metric	HDFS (%)	HDFS-DQN (%)
Performance under standard conditions	85	90
Performance under varying conditions	70	85
Adaptability (A)	$\frac{70}{85} = 0.82$	$\frac{85}{90} = 0.94$

Table 8: Energy consumption comparison

Metric	HDFS	HDFS-DQN
Power consumption rate ($P_{encrypted}$)	50 W	40 W
Time taken for encryption ($T_{encrypted}$)	120 ms	100 ms
Energy consumption (E)	$50 \times 0.12 =$ 6 J	$40 \times 0.1 =$ 4 J

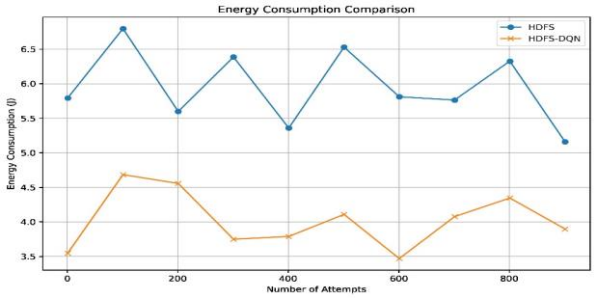

Fig. 13: Energy consumption line plot

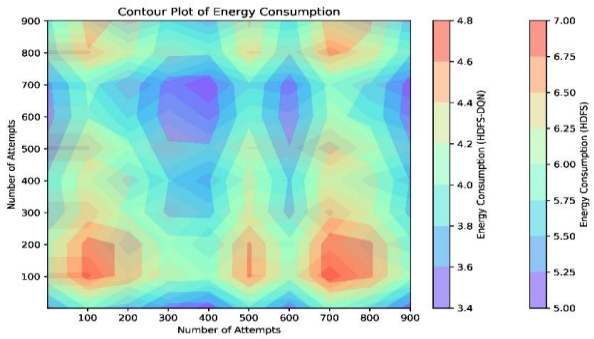

Fig. 14: Energy consumption contour plot

Discussion

The recommended results from our flexible analysis clearly depict the fact that the proposed DRL-based encryption model that involves DQN is significantly superior to the existing HDFS model. Therefore, each evaluation metric establishes that the HDFS-DQN model surpasses the others and proves that it can improve data security and system performance. In the next sections, we report the results of the learning analytics in their entirety for each of the metrics selected.

Security Level (S)

The security level is the standard that defines the efficiency of the system at times when the amasses attacks. It is defined as the ratio of broken exploits to the overall number of exploit attempts that were carried out. The safety extent can be defined by the ratio of the size of an organism to its predators; the lower figure means a higher security level.

The best security level was described to be 0 in the employment of the conventional HDFS model. 05 is the success rate with fifty attack targets successfully compromised out of 1000 attempts. On the other hand, the HDFS-DQN model was able to earn a security level of 0 which is quite a contrast of the traditional model. 01, which is equal to 10/ 1000, meaning a very low success rate of 1 percent. The decrease in the rate of successful attacks to a mere 13% points to the

effectiveness of the security features of the HDFS-DQN model shown in Table (3). This adaptive control can be attributed to the integration of DQN to the DRL where the encryption strategy is adapted near real-time in accordance with the access and threat patterns hence mitigating the vulnerabilities and enhancing the security of the overall system.

Computational Overhead (C)

When discussing computational overhead one has to identify the load in terms of time which is required to perform encryption in comparison with the point of time which is needed without the encryption. This metric is crucial for monitoring the performance of the encryption algorithms where more computational overhead equates to the system's slower running.

When considering Table (4) the results of our evaluation, we find that the traditional HDFS model has a computational overhead of 40 ms and encrypted processing took 120 ms than the baseline of 80 ms when using HDFS-DQN we pointed out that there would be a significant cut down on computational overhead because it only takes 20 ms when processing the overhead and 100 ms when processing the encrypted values. This way, the DQN-enhanced DRL integration allows the system to update encryption parameters in the process dynamically and makes the process much faster compared to when it has to encrypt it on its own.

Latency (L)

Whereas, throughput introduced the concept of latency, which is the additional amount of time taken to encrypt data when it is being accessed. This is an important criterion for processing real-time data and its quality and usability for end-users.

In the conventional HDFS model shown in Table (5), latency was recorded to be around 150 ms while in the HDFS-DQN model latency varied around 110 ms; thus, reducing latency by 40 ms needs a good strategy to be applied and this was made possible by the DQN-enhanced DRL approach that minimized delays caused by encryption. The capability of the HDFS-DQN model in constantly studying and training based on the changing access data patterns means that the model enhances the encryption process to increase the rate of data retrieval and reduce the time hence increasing user satisfaction.

Resource Utilization (U)

Resource utilization aims to determine the number of computational resources that are employed while encrypting. Resource utilization must hence be optimized to ensure that an increase in productivity is realized and that scalability of the systems is achieved.

In the traditional HDFS model entirely 60% resources were dedicated for the encryption, where in the HDFS-DQN model only 45% were dedicated for the same. This cuts down the resource that is required especially in a physical setting hence creating efficiency in the proposed model shown in Table (6). To address the above problem, the HDFS-DQN model adjusts encryption parameters and dynamically allocates resources to keep the encrypted processes within system limits the proposed HDFS-DQN model increases the system performance and scalability by setting optimal encryption parameters and managing the system resources.

Adaptability (A)

Regarding adaptability, this refers to how well the encryption algorithm can perform amidst fluctuating data access patterns and threat levels. It is a valuable measure when determining how well-prepared and adaptable that strategy of encrypting is.

Thus, the HDFS-DQN model appears far more adaptable than the baseline model; the percentage of performance metrics more accurately represented as a score yields an adaptability figure of 0.94 compared to 0.82 in the case of the original HDFS model of computing. This means that the HDFS-DQN agent has the ability to determine fluctuating conditions in order to optimize on encryption procedures of HDFS to ensure optimum security and performance shown in Table (7). The reinforcement learning contour enables the model to self-devel over data access patterns and threats experienced and improve its encryption method accordingly.

Energy Consumption (E)

Energy consumption measures how much power is used to achieve the functionality of the mathematical algorithms for encryption and decryption. Thus, the consumption of energy is desirable by reducing the overall expenses as well as environmental impact.

The last indicator shown in Table (8) was the energy consumption of the HDFS model and the HDFS-DQN model differently. The HDFS consumed 6 J for encryption and the proposed HDFS-DQN consumed 4 J; that is, the HDFS-DQN consumed a third less energy compared to the traditional HDFS. In this respect, the computational time and energy consumption in HDFS-DQN is reduced, besides enhancing the performance by optimizing the values of the encryption parameters necessary for the functioning of large-scale distributed storage systems.

Significance of Proposed Model

The achievement of HDFS-DQN with multiple aspects of every index is higher than the comparative HDFS, it shows that HDFS-DQN is superior to HDFS. The proposed DQN-integrated DRL approach improves security, its efficiency with less overhead, relatively low latency, better resource utilization, better adaptability and

low energy requirement. All the improvements highlighted above prove that the increase in the extent of reinforcement learning makes it possible to use new methods in the development of encryption tactics for distributed storage systems. This research opens new avenues for more investigations in intelligent encryption systems, hence providing a solid and optimized framework to protect LSM-DS storage architectures. The proposed security model is scalable and highly adaptive to the change in threat patterns and data access; making it possible to fit current and future datasets storage systems.

Scalability and Limitations

In an HDFS environment, with fluctuating traffic and access patterns, several aspects are examined. First, HDFS grows then the capability of encryption may face limitations such as latency and resource demand due to an increase the computational overhead. While DQN enhanced DRL approach adapts encryption in real time it might still experience performance degradation when managing large and complex distributed environments across nodes. Furthermore, in high-traffic cases, the model faces latency issues if the model adaptive framework fails to balance between rapid data access and robust security requirements. The solution of this case involves optimal reinforcement learning parameters and includes multi-agent reinforcement parameters to distribute the decision-making nodes across multi-agents. Furthermore, using hierarchical adaptive encryption techniques would allow for variable safety forces across different layers of HDFS, enhancing scalability and performance without compromising security. This balance integrates the model in complex and real-world distributed environments.

Conclusion

Therefore, this research proposes an innovative encryption model for HDFS using DRL with DQN enhancement to boost the securities against the conventional model and improve the computational time, latency, resource usage, flexibility and power profile of the Hadoop environment. When the values and accesses are reflected in the data the model can dynamically and efficiently change the encryption parameters thus offering proper security to the data. However, the study is not without its drawbacks; the first is that such an advanced technique may not be easily implemented in most existing systems and the second is the computationally intensive process involved in the training step. For future work, development must be made in fine-tuning the model to minimize the computational load as much as possible; the scalability of the approach in various data settings must be explored; and different flavors of machine learning paradigms must be incorporated to make it more responsive. Additionally, authors simulated the development of the proposed Hybrid AI to enhance scalability and accuracy, switch to unsupervised learning and federating learning solutions across distributed nodes and encryptions to self-adjust in response to real-time threat assessments. Furthermore, authors are encouraged to conduct more studies to identify possible weaknesses and countermeasures to enhance the random nature of the applied cryptography technique to withstand other cyber threats.

Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

Shivani Awasthi: Conceived and designed the experiments, performed the experiments, performed the computation work, analyzed the results, prepared all figures and/or tables, and prepared all the drafts and the final manuscript.

Narendra Kohli: Supervised the entire work and approved the final draft.

Ethics

The authors confirm that this manuscript has not been published elsewhere and that no ethical issues are involved.

References

Al Jallad, K., Aljnidi, M., & Desouki, M. S. (2019). Big data analysis and distributed deep learning for next-generation intrusion detection system optimization. *Journal of Big Data*, *6*(1), 88. https://doi.org/10.1186/s40537-019-0248-6

Alhazmi, H. E., & Eassa, F. E. (2022). BCSM: A BlockChain-based Security Manager for Big Data. *International Journal of Advanced Computer Science and Applications*, *13*(3). https://doi.org/10.14569/ijacsa.2022.0130364

Alpaydin, E. (2020). Introduction to Machine Learning, 4th Ed.

Anand, A., & Hassabnis, A. (2024). Qcrypt: Leveraging Post-Quantum Cryptography for Enhanced Security of Data at Rest. *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–9. https://doi.org/10.1109/icccnt61001.2024.10725265

Belhadaoui, H., Filali, R., & Malassé, O. (2023). A Role-Attribute Based Access Control Model for Dynamic Access Control in Hadoop Ecosystem. *IAENG International Journal of Computer Science*, *50*(1).

Du, H., Han, P., Xiang, Q., & Huang, S. (2020). MonkeyKing: Adaptive Parameter Tuning on Big Data Platforms with Deep Reinforcement Learning. *Big Data*, *8*(4), 270–290. https://doi.org/10.1089/big.2019.0123

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file. Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, 29–43. https://doi.org/10.1145/945445.945450

Hamza, R., Hassan, A., Ali, A., Bashir, M. B., Alqhtani, S. M., Tawfeeg, T. M., & Yousif, A. (2022). Towards Secure Big Data Analysis via Fully Homomorphic Encryption Algorithms. *Entropy*, *24*(4), 519. https://doi.org/10.3390/e24040519

Khandagale, S. B., Narain, B., & Jadhav, B. T. (2024). Enhancing Big Data Security in Hadoop using Machine Learning. *International Journal of Scientific Research in Science, Engineering and Technology*, *11*(6), 304–309. https://doi.org/10.32628/ijsrset24116182

Li, Y., Chen, R., & Rahmani, R. (2023). Secure Data Sharing in Internet of Vehicles Based on Blockchain and Attribute-Based Encryption. *2023 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 56–63. https://doi.org/10.1109/smartiot58732.2023.00016

Mashonganyika, F., Chibaya, C., & Rupere, T. (2020). Real-Time Self-Adaption of Network Security Mechanisms for Dependable Distributed Systems. *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 1–7. https://doi.org/10.1109/imitec50163.2020.9334148

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Mustafa, I., Khan, I. U., Aslam, S., Sajid, A., Mohsin, S. M., Awais, M., & Qureshi, M. B. (2020). A Lightweight Post-Quantum Lattice-Based RSA for Secure Communications. *IEEE Access*, *8*, 99273–99285. https://doi.org/10.1109/access.2020.2995801

Nenov, L. (2024). Reinforcement Learning for Key Management in Distributed Systems. *2024 32nd National Conference with International Participation (TELECOM)*, 1–5. https://doi.org/10.1109/TELECOM63374.2024.10812245

Nijil Raj, N., Rajesh, R., Justin, A., & Shihab, F. (2024). Enhancing Network Intrusion Detection Using Deep Reinforcement Learning: An Adaptive Learning Approach. *Proceedings of the Second International Conference on Computing, Communication, Security and Intelligent Systems*, 297–315. https://doi.org/10.1007/978-981-99-8398-8_21

Olaoluwa, F., & Potter, K. (2024). Deep Learning for Intrusion Detection Systems (IDS). *Preprints*. https://doi.org/10.20944/preprints202409.0411.v1

Pronika, & Tyagi, S. S. (2021). Secure Data Storage in Cloud using Encryption Algorithm. *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 136–141. https://doi.org/10.1109/icicv50876.2021.9388388

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 1–10. https://doi.org/10.1109/msst.2010.5496972

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Singh, S., Jasim, L. H. D., Dhote, V., Suseela, D., & Venkatasubramanian, R. (2023). Privacy-Preserving Data Mining Methods for Sensitive Information. *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 841–844. https://doi.org/10.1109/ictacs59847.2023.10390087

Sood, S. K., Sarje, A. K., & Singh, K. (2011). A secure dynamic identity-based authentication protocol for multi-server architecture. *Journal of Network and Computer Applications*, *34*(2), 609–618. https://doi.org/10.1016/j.jnca.2010.11.011

Sunder, A., Shabu, N., & Remya Nair, T. (2022). Securing Big Data in Hadoop Using Hybrid Encryption. *Ubiquitous Intelligent Systems*, 521–530. https://doi.org/10.1007/978-981-16-3675-2_39

Tabbassum, A., & Abdul Kareem, S. (2021). Implementing Zero Trust Security Models in Cloud Infrastructures. *International Journal of Science and Research (IJSR), 10*(11), 1582–1586. https://doi.org/10.21275/sr211110212612

Wang, H., Wang, Q., Ding, Y., Tang, S., & Wang, Y. (2024). Privacy-preserving federated learning based on partial low-quality data. *Journal of Cloud Computing*, *13*(1), 62. https://doi.org/10.1186/s13677-024-00618-8

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*, 1995–2003.

Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J., & Lv, W. (2019). Edge Computing Security: State of the Art and Challenges. *Proceedings of the IEEE, 107*(8), 1608–1631. https://doi.org/10.1109/jproc.2019.2918437