Journal of Engineering Science and Technology
Vol. 16, No. 5 (2021) 3772 – 3791

A NEW ENCRYPTION SCHEME FOR PERFORMANCE IMPROVEMENT IN BIG DATA ENVIRONMENT USING MAPREDUCE

THOYAZAN SULTAN ALGARADI*, B. RAMA

Department of Computer Science, Kakatiya University, Warangal, India
*Corresponding Author : yaz.sul77@gmail.com

Abstract

There are diverse methods and technologies adopted by authors studied to Keep up with the rapid progress of big data technology in securing big data in the field of cryptography, and that seeks to bring the latest innovative solutions to address the problems related to severe threats to the security of sensitive information on the big data environment. Currently, this area includes many standard encryption algorithms to cover the part of the protection of big data in a specific environment as those provided an excellent performance that contributed to the progress and development of this area. Yet, these methods still have some major limitation related to its performance, throughput, execution time, and many other. Such a move was as a starting point for researchers to work, develop, and try to come out with positive results. In this paper, we studied many of these works and raised with our cryptographic algorithm that meets the purpose. As a result, we overcome most of the weaknesses which the area was faced previously and come up with the best results. Besides, the proposed scheme ensures a high level of security where its performance analysis proves the effectiveness and superiority in comparison, which makes a fair balance between security, time-consuming, and performance. Moreover, we used a pre-configured programming model of MapReduce to apply the proposed and deal with large amounts of data and encrypt them in a parallel way that ensures a shorter processing time.

Keywords: Big data, Blowfish, Cryptography, Encryption algorithm, Security analysis, MapReduce.

1. Introduction

The daily extracted data in the present day are increasing unreasonably, making the need for the emergence of the term big data come [1]. To control and make these data meaningful, we dealt with them by applying a particular technical analysis introduced mainly to cope with such data. These data usually cannot be processed, stored, searched within, and analysed using conventional database management tools and traditional database management systems [2]. Moreover, what distinguishes big data techniques is its ability to deal with enormous volumes of structured and unstructured data that are huge [3], where we can determine its characteristics as the properties like Volume, Variety, Velocity, Variability, Complexity [4].

Big data security includes computation and database operations for the massive amount of the various data placed away from the data owner's enterprise. It recognizes that the purpose of the big data environment is to access and get the required data from different areas; this is an underlying assumption that must be achieved. Hence, security will play a significant role, as one of the serious problems and issues that must be studied well in the domain of big data and arrive at the best research that can cover this Imperfection [5].

Data security and privacy are two of the most influential aspects in the big data field [6], which takes care of how users authenticated to ensure more reliable communication, and then encrypt and send data and how to store it in a way that makes it difficult for hackers to access [7]. Judging from the above, big data security has become a key issue as a field of research and diving to study vulnerabilities and try to find the best solutions [8]. In the term of information security, many extensible algorithms specialize in the field of encryption that has shown great success through its usage in the science of information and communication [8, 9]. Where the more significant the complexity of the encryption algorithm, the more excellent its resistance against hacking attempts and the success of the hacking attempt by hackers, which formed the rule that greater complexity means greater security [10]. These algorithms can be symmetric key algorithms that employ a single key for both encryption and decryption and are known as a secret key algorithm or maybe asymmetric key algorithms that uses a public key and private key pair which are related to each other, and this known as public-key algorithms [11].

As shown in Fig. 1, several systems have proven efficient in keeping with all data security requirements in the field of big data encryption, such as the high data generation speed, the size of those data, and the level of their security. RC6, MARS, 3DES, DES, and Blowfish are some of the symmetric encryption techniques which tried to offer comparatively effective solutions and enveloped most of the issues that emerged in the field of security. However, these algorithms are no longer satisfactory enough to achieve efficient results consistent with the characteristics required for processing big data [12] due to the challenges related to the variety of data and the speed of its generation and its volume, which are doubling day by day creating a challenging environment to handle it using traditional techniques [13]. For that, an important platform used for this purpose is called Hadoop. It is a well-known and widely used open-source implementation of the MapReduce framework in big data [14] as it uses Hadoop distributed file system (HDFS) to stores a massive amount of data by dealing with MapReduce as a programming model to efficiently

process data by generating large data sets on a cluster with a parallel and distributed algorithm [9]. That is useful for large, long-running jobs that cannot handle within the scope of a single request. This work will be split into several blocks by a mapper, and the reducer will merge those blocks to get a single output. Besides, it produces and stores three copies of each block to ensure fault tolerance. Thus, for dealing with a big data environment, most of the researchers adopted Hadoop as an open-source framework to apply their solutions using the MapReduce programming model by java execution [14-16].

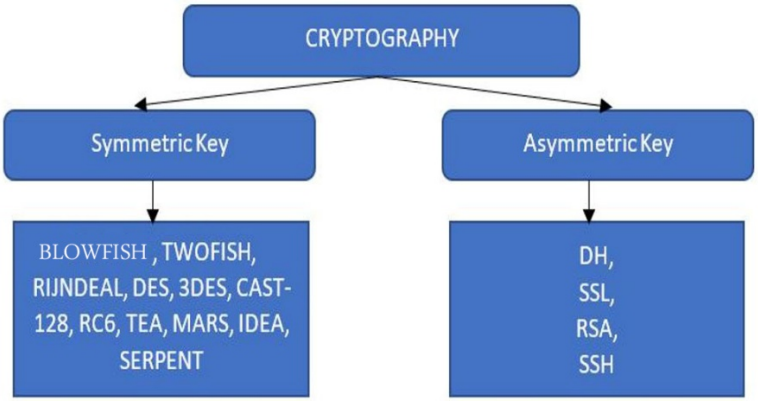

Fig. 1. The main categories of cryptography [17].

We organized the remaining parts of this paper as follows: we give a literature review of the current field works in Section two. In Section three, we introduce the proposed scheme, the present the result and discussion in Section four, the experiment setup presented in Section five, and finally, the conclusion drowned in Section six.

2. Literature Review

Today's big data is a vast area for research and work because of the challenges of data privacy and security that is a significant concern in big data analytics, thus makes it occupy a large space accelerated towards researchers. Encryption, privacy, communication, authentication, integrity, and access are among the different levels of big data security. Data encryption is a significant concern concerning data security and privacy in a big data environment.

After looking at many of the proposed papers, we found that one of the most comfortable algorithms with high efficiency than some counterparts is the blowfish algorithm. It considers as one of the strong and completely unbreakable encryption algorithms to date with a convenient design for any enhancement or modification in its structure [18]. It has a greater throughput than 3DES, DES, CAST-128, IDEA, and RC2 with better performance and efficiency for analysing Encryption/Decryption time and throughput and for the power consumption value where they concluded that blowfish. Kofahi et al. [19] show that the blowfish algorithm is the fastest, followed by the DES algorithm then the T-DES algorithm. Moreover, blowfish need the least amount of time for the file to encode and minimum time for text file decode compared to the AES algorithm [20].

Algarad and Rama [6], proposed a novel blowfish-based algorithm, which is an improved algorithm of the standard blowfish algorithm, where they ensured a high level of security. Their proposed used a pre-configured programming model of MapReduce to deal with a large amount of data and encrypt them in a parallel way, then proved its efficiency by achieving shorter processing time. As we observed, the scheme's results to encrypt and decrypt data increased the power and safe consumption by about more than 50%, where the percentage increased whenever the size of the file increased. Thus, it proved a fair balance between security and time-consuming performance. Therefore, Desai et al. [13] used MapReduce too to propose a framework that reduces encryption costs and decreases the time and makes it more efficient. It can boost encryption performance by encrypting a large amount of data in a parallel and distributed fashion. They enhanced the encryption performance in the term of execution time by using MapReduce and with minimal usage of system resources. Their work was good for better results, but the drawback may be on an AES, where it has certain limitations in case of security and scalability, and it takes extra time, as well the differences between methods increase with file size.

Furthermore, Jayan and Upadhyay [21] proposed an enhancement work using MapReduce, which is quite similar to that [13] that used a modified parallel RC4 encryption algorithm [22]. Their work has shown positive results by improving and enhancing big data security and algorithm cost, which reduced them using MapReduce. They show that AES was an effective algorithm but takes more time, the RC4 took a shorter time but was sequential, and the modified RC4 was the best effect with MapReduce. In the same field and to enhance the security of essential data at the HDFS level, Kadre and Chaturvedi [9] provided AES-MR as a new solution for a data security issue of encryption at the storage level at HDFS. They show that their work with the help of parallel processing of map and reduced function was fast enough to encrypt the data chunks from the data store, but their work fell into trouble due to the increased size of data files, size of log files (files which record the data activity), intrusion detection system faced so many problems and gave inaccurate results for detecting the attackers [3].

In this paper, we have proposed a durable approach to data security in big data environment encryption and transmission using Hadoop cluster where the complete encryption process is done smoothly, with ensures complex data concealment by creates a problematic pattern that is difficult to break during intentional attacks while being transferred in a big data environment. We used the MapReduce framework as a programming model for encrypting large amounts of data in parallel and distributed fashion by using our proposed approach for the encryption scheme to perform the encryption parallel using the Hadoop cluster. Thus, we reduce the cost of encryption where MapReduce and a key generated during the proposed encryption scheme. In other words, we tried to mimic data cryptography while transferring data from the user to HDFS and how to store it in a practical way that ensures its security far from the detection of any attempt to target it.

MapReduce, a pre-configured programming model, was designed to deal with large amounts of data and encrypt it in a distributed and parallel way to ensure the processes' speed. Thus, we will overcome the most prominent obstacles because the data will be encrypted by applying for our proposed program, then by using the Hadoop cluster and with the help of MapReduce, the cost of encryption will be reduced more. The framework will divide the big file into small pieces called

blocks. The data of each will be encrypted using the proposed algorithm then create three copies of each before storing it, which will be useful to ensure tolerance fault. The Key will send as a key-value pair to the reducer. The rest of the operations will work as the normal MapReduce.

3. The Proposed Scheme

We can understand the proposed scheme idea by the general structure that presented in Fig. 2, which it illustrates the adopted scenario as follows:

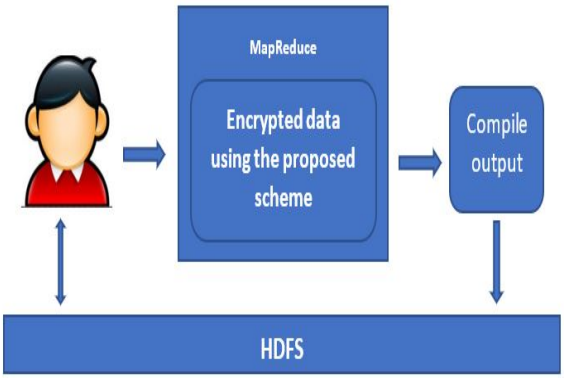

Fig. 2. The general structure of the proposed approach.

The proposed formulated in two phases. First: explain the proposed method that facilitates the continent to understand and know how it works, then lists the algorithm steps and shows an illustration figure to understanding more. Second: is how the proposed scheme work using MapReduce.

3.1. The proposed algorithm

Through this paper, we proposed a new scheme for encryption and mixed it with the MapReduce parallel programming paradigm. It is considered as one of the best, quickest conceivable approaches to process and executes the encryption of the data faster than previously while maintaining efficiency and improve it. The proposed processes an enormous volume of data by processing a large size block of 384 bits in a complex and rigorous way that ensures the difficulty of breaking it, using an unspecified encryption key can range between 128-896 bits, which prove the strength of the proposed and make it difficult for attackers to break it.

The proposed will work to divide the main block into three equal size blocks, each of 128 bits, then uses the function (F) to ensure the change of bits position ultimately. This function applies to the middle block with swapping the location of the three blocks in each round. That ensures the total change using a derivative encryption key from the master key that ranges between 128-896 bits applied in each round to change the bits' value. The number of rounds used is eight rounds, which are accurately set to ensure the total position change between the three initial blocks and then ensure the change of bit locations within each block and change its value. Figure 3(a) shows the change that happened for positions of the main three blocks derived from the main block where Fig. 3(b) shows the change happened

for positions inside each block. Figure 3(c), on the other hand, shows the change that happened for each bit.

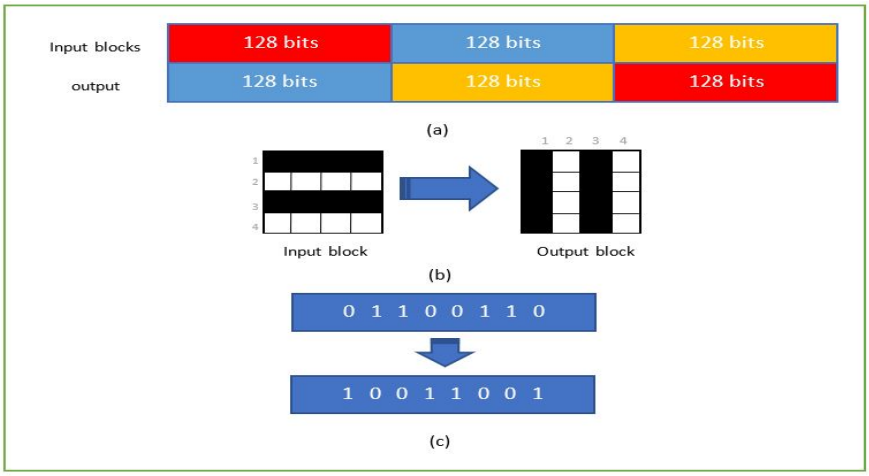

Fig. 3. An illustration of the resulting form of the processed block.

3.1.1. Keys derivation procedure

P-array of the proposed algorithm is an eight-element array, and each element has a 128-bits size that may look as *P= {P1, P2, P3, ..., P8}*. The proposed scheme uses one element for encryption each round. As the proposed algorithm deals with the main Key of 128-896 bits size, so by iteration, we can generate the K-array, which will have eight elements, each of 128 bits. Thus, we can derive the p-array as the following steps:

- Enter the main Key.
- Generate K-array from the main Key with eight elements, each of 128 bits.
- Generate P-array with the same size of K-arrays and derived its elements as follows:

$For\ (i = 0\ to\ i = 7)$

$p[i] = p[i]\ XOR\ k[i]$

3.1.2. Function F

This function takes a 128-bits input, which expresses the middle part of the main block that we process. Consider that this part as a matrix of 4×4 element, each element contains 8 Bits as following:

$i11$	$i12$	$i13$	$i14$
$i21$	$i22$	$i23$	$i24$
$i31$	$i32$	$i33$	$i34$
$i41$	$i42$	$i43$	$i44$

The function is to work in two steps:

- Convert rows to columns and columns to rows.

$i11$	$i21$	$i31$	$i41$
$i12$	$i22$	$i32$	$i42$
$i13$	$i23$	$i33$	$i43$
$i14$	$i24$	$i34$	$i44$

- For each element, convert 1 to 0, And 0 to 1.

For example:

$i11$= 01100110

$it\ will\ be$

$i11$=10011001

Note: in the real implementation, we will deal with the block like a regular array of 16 elements each of 8-bits, and we simply used the matrix in the explanation to just easily understand the function work manner.

3.1.3. The proposed algorithm steps

The idea that the proposed include is to create an efficient algorithm for encryption and make it able to handle large size of a block using a rigorous manner, and that will ensure the strength of the efficiency of the output and make complexity greater to verify the impossibility of breaking it. We have formulated the proposed algorithm through the following steps:

1. **Take X as a block of the size 348 bits.**
2. **Split X to XL, XM, and XR each with the size of 128 bits.**
3. **For the first round (i=1) Do:**
   XL=XL XOR Ki
   XM= f (XM) XOR XL
   XR=XR XOR XM
   As the value of (i) is an odd number, swap XL and XM.
4. **For the second round (i=2) Do:**
   XL=XL XOR Ki
   XM = f (XM) XOR XL
   XR= XR XOR XM
   As the value of (i) is a double, swap XM and XR.
5. **Repeat steps 3 and 4 till i=8.**
6. **Do:**
   XM = f (XM)
7. **Recombine XL, XM, and XR.**

3.2. The proposed scheme using MapReduce

Hadoop is a very famous and widely used open-source implementation of the MapReduce [8] framework. The Hadoop distributed file system (HDFS) stores a large amount of data. The mapper will divide the work into blocks, and the reducer will merge the blocks producing a single output. For that, the proposed encryption scheme is used for the encryption to perform the encryption parallel using MapReduce of the Hadoop cluster. Where the parallel encryption processes of MapReduce can certainly improve performance and reduce the cost of encryption because instead of encrypting blocks one by one, multiple mappers can work on encrypting different blocks [10]. After that, the reducer combines all blocks and store them back in HDFS. For more clarify, the framework will split and mapped the big files into several small blocks. Then produced and stored three copies of each block to ensure fault tolerance. Before that, the data have to be encrypted using the proposed scheme. Then the Key is sent as a key-value pair to the reducer. All other process is the same as the normal MapReduce. Figure 4 clarifies and simplifies the understanding of the above steps of the proposed.

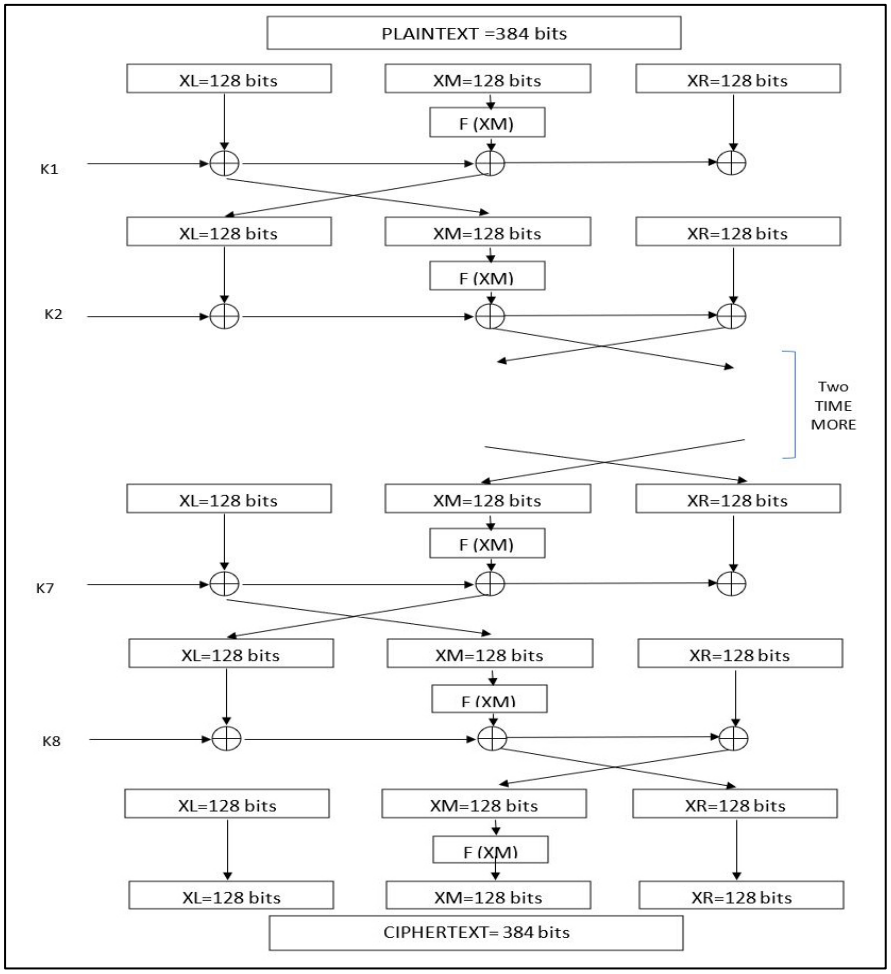

Fig. 4. The block diagram of the proposed scheme.

4. Results and Discussion

In this section, we evaluate and discuss the proposed algorithm and its advantages using the following factor:

4.1. Objective evaluation metrics

The proposed method experiments have been conducted on a random text file using a random key. There are many scientific metrics of objective evaluation to measure the imperceptibility between the original text and the encrypted text, as the original text of the proposed scheme is binary. Thus, the objective evaluation [23, 24] is applying with some differences, where will apply based on mutual relations between the input bits in the original text file and its corresponding bit in the encrypted text file. Hence, it fairly suits binary array.

Here, error metrics used for calculating the objective evaluation concerning the original text file and its encryption one. To calculate that, firstly, we have to declare *TP, FP, TN*, and *FN* as a total number of true-positive, false-positive, true-negative, and false-negative bits concerning the original text file and its encryption, respectively.

i. Recall/Sensitivity

$$Recall = \frac{TP}{TP+FN} \tag{1}$$

The ideal value of sensitivity is one that closes to 1.

ii. Precision

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

The ideal value of precision is one that closes to 1.

iii. Specificity

$$Specificity = \frac{TN}{TN+FP} \tag{3}$$

The ideal value of specificity is one that closes to 1.

iv. Accuracy

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{4}$$

The ideal value of Accuracy is one that closes to 1.

v. F-Measure

$$FM = \frac{2\times Recall\times Precision}{Recall+Precision} \tag{5}$$

The ideal value of F-measure is one that closes to 1.

vi. Negative Rate Matrix (*NRM*)

$$NRM = \frac{R_{FN}+R_{FP}}{2} \tag{6}$$

where $R_{FN} = \frac{FN}{FN+TP}$, $R_{FP} = \frac{FP}{FP+TN}$ Moreover, the outstanding value of F-measure is one that closes to 0.

vii. Balanced Classification Rate (*BCR*)

$$BCR = 0.5 \times (Specificity + Recall) \quad (7)$$

The ideal value of BCR is one that closes to 1.

viii. Balance Error Rate (*BER*)

$$BER = 100 \times (1 - BCR) \quad (8)$$

The ideal value of *BER* is one that closes to 0.

Table 1 shows the objective evaluation metrics for the current scheme, the novel blowfish-based algorithm scheme [6], and the standard blowfish. We calculated the results by a random text file and its corresponding encrypted text file. As noticed, all the measured objective metrics of the proposed scheme are approaching their ideal values more than they are in the other two algorithms, which prove the effectiveness of the proposed scheme.

Tables 1 has graphically drawn in Fig. 5 It represents the imperceptibility measurements between the normal blowfish, the novel blowfish, and the proposed scheme.

Table 1. Objective evaluation metrics.

	Blowfish	Novel blowfish [6]	The proposed	Ideal value
Recall	0.52427	0.65957	0.82429	1
Precision	0.26470	0.60784	0.85980	1
Specificity	0.46619	0.59183	0.81176	1
Accuracy	0.48177	0.62500	0.81875	1
F-Measure	0.35179	0.63265	0.84162	1
NRM	0.50476	0.37429	0.18196	0
BCR	0.49523	0.62570	0.81803	1
BER	50.47679	37.42943	18.19681	0

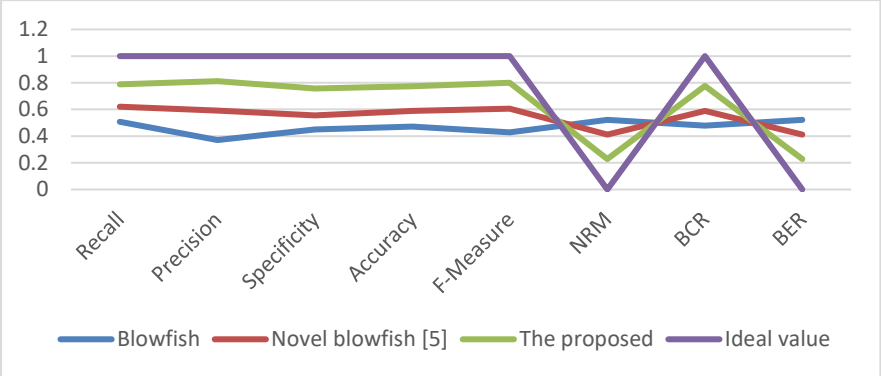

Fig. 5. Graphical representation of imperceptibility measurements.

4.2. Statistical analysis

4.2.1. Entropy

To evaluate the text quantity is called entropy and consider as a measure to estimate the amount of randomness [25], $H(S)$ is expressed mathematically for the message m as follows:

$$H(S) = \sum_{i=0}^{n-1} P(S_i) log_2 \frac{1}{P(S_i)} \quad (9)$$

The probability symbol S_i is represented as $P(S_i)$, and because there are 256 possible values for the message, the ideal value is $H(S)$=8. Hence, the highly random of the ciphertext in nature is the value that is approaching 8. Figure 6 illustrates that the proposed provides satisfying results that come close to the ideal value and make it better.

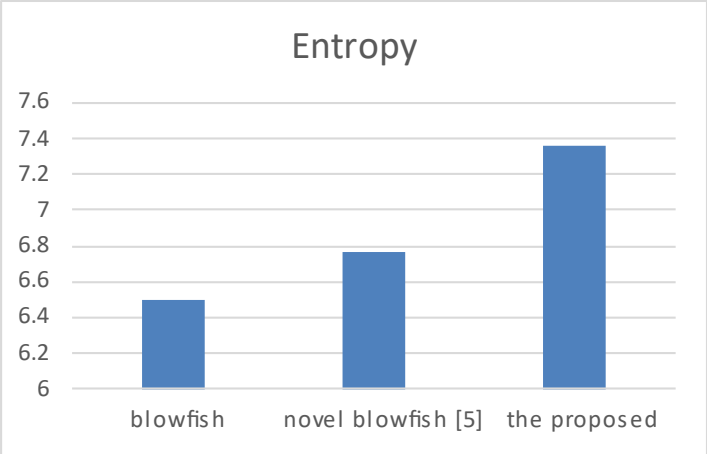

Fig. 6. graphical representation of the entropy analysis.

4.2.2. Variance

Variance knew as the measure of how far each data set value from the mean. it is symbolized as (σ^2). Its formula can mathematically express as follows:

$$\sigma^2 = \frac{\Sigma(X-\mu)^2}{N} \quad (11)$$

where μ refers to the mean values of X, and N refers to the number of terms in the distribution.

4.2.3. Standard Deviation (*SD*)

It refers to the measurement of how spread-out numbers are. It is calculated as the square root of the variance and symbolized as σ. its formula mathematically expressed as follows [26]:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \quad (12)$$

4.2.4. Mean-to- Standard Deviation (*MSDR*)

This measurement illustrates the statistical data points dispersion measurement in a data series around the mean. It is known as the coefficient of variation (*CV*), and it represents the ratio of the standard deviation to the mean. Its formula mathematically expressed as follows [27]:

$$CV = \frac{\sigma}{\mu} \tag{13}$$

Table 2 shows the results obtained related to Mean, Variance, Standard Deviation, and Mean-to- Standard Deviation of the proposed scheme. The results are identically similar to the original text and recovered text as measured by three experiments of different size files using the proposed scheme, which means that the statistical analysis proves that all algorithms are efficient due to their ability to recover the original text without any loss of information.

Table 2. Statistical analysis (SD and MSDR).

Experiment		Statistical Analysis			
		Mean	**Variance**	**SD**	**MSDR**
1st experiment	**Original text**	68.5	5.25	2.2912	0.0334
	Recovered text	68.5	5.25	2.2912	0.0334
2nd experiment	**Original text**	76.5	5.25	2.2912	0.0299
	Recovered text	76.5	5.25	2.2912	0.0299
3rd experiment	**Original text**	84.5	5.25	2.2912	0.0271
	Recovered text	84.5	5.25	2.2912	0.0271

4.3. Visual assessment

This measurement finds out if a hacker can know or imagine some useful information by visual assessment of the ciphertext that may indicate the similarity between the input and the output.

Figure 7 shows the encrypted output, which proves the difficulty for attackers to infer and get even some meaningful information by visually assessing the result. That means the proposed is booming due to no slightest similarity relationship between the plaintext and ciphertext.

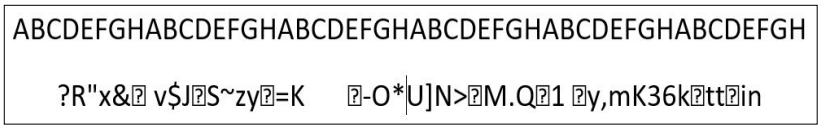

Fig. 7. A form of data before and after being encrypted by the proposed algorithms.

4.4. Security analysis

In this section, we provide a security analysis of the proposed scheme compared to the standard blowfish and the novel blowfish [6]. The sides of the comparison areas the following Table 3:

Table 3. Comparison of the security analysis based on the algorithm's architectures.

Security algorithms	Cipher type	Key size	Block size	Keyspace	Round
Blowfish	Symmetric block cipher	(32 – 448) bits	64 bits	$(2^{32} - 2^{448})$	16
Novel blowfish [6]	Symmetric block cipher	(64 – 448) bits	128 bits	$(2^{64} - 2^{448})$	7
The proposed method	Symmetric block cipher	(64 – 512) bits	128 bits	$(2^{128} - 2^{896})$	8

4.4.1. Keyspace (Brute-force-attack)

It refers to the measurement of all keys possible permutation, set to ensure whether the cryptographic algorithm more efficient and sensitive to simple change of the Key used during the encryption results, thus ensure the resistance of the algorithm toward brute-force-attacks. Here, the keyspace is large enough to make an adversary search infeasible. As on average, the attacker must search in half the keyspace to find the solution [28].

Table 3 shows the key size and keyspace of the standard blowfish, novel blowfish [6], and the proposed scheme. The values mentioned show the efficiency of the proposed algorithm when compared to the rest. That is clear as we provided the maximum complexity to find the correct Key as its value can be appreciated to be close to 5.282945e+269.

4.4.2. Key sensitivity analysis

Key sensitivity analysis is a measurement to check the encryption algorithm's sensitivity towards change in initial conditions. Here, a suitable algorithm produces an entirely different ciphertext than the previous if it is slightly changed in its encryption key.

In the proposed algorithm, as mentioned in the section of the Key Derivation Procedure, we derived the used p-array from the general k-array that we created by using the main Key. That happens by using the XOR operation to generate each round Key. It proves that changing all rounds keys is an inevitable result if any slight change happened on the main Key, which in turn will produce an entirely different ciphertext.

4.5. Correlation analysis

It is calculated among the encrypted data and the original data to analyzes the confusion properties of encrypted data as a part of statistical analysis and shows the relationship between them. This analysis should come with results that are close to zero as to be in correlation ideally. We can use the following formula to calculate that for the plaintext.

$$r\alpha\beta = \frac{cov(\alpha,\beta)}{\sqrt{D(\alpha)}\sqrt{D(\beta)}} \quad (14)$$

$$cov(\alpha,\beta) = \frac{1}{N}\sum_{i=1}^{N}(\alpha_i - E(\alpha))(\beta_i - E(\beta)) \quad (15)$$

$$D(\alpha) = \frac{1}{N}\sum_{i=1}^{N}(\alpha_i - E(\alpha))^2 \quad (16)$$

$$D(\beta) = \frac{1}{N}\sum_{i=1}^{N}(\beta_i - E(\beta))^2 \quad (17)$$

Moreover, α and β denote two values for which correlation needs to calculate. N refers to the total elements number obtained from the data, whereas $E(\alpha)$ = mean of α, and $E(\beta)$= mean of β.

Figure 8 shows the results obtained, in which the smallest correlation coefficient value that is approaching the value of 0 has achieved in the proposed scheme as well, as the compared algorithms came with good results.

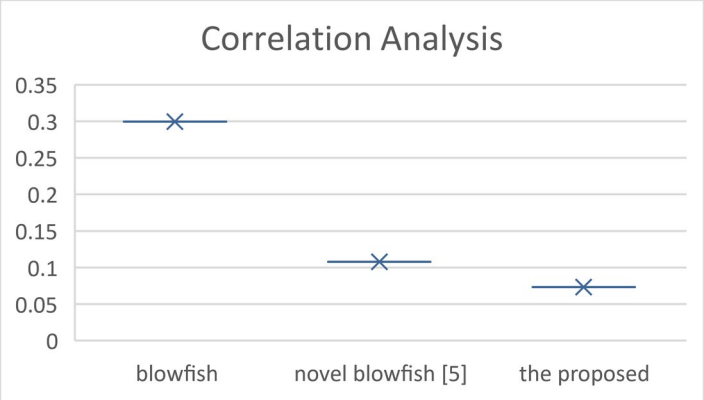

Fig. 8. Graphical representation of the correlation analysis.

5. Experiment Setup

This section will present the proposed scheme performance evaluation through its implementation and the results obtained by using MapReduce and without using MapReduce, and then results will be compared with some existed works. We rebalanced the load for distributed file system through simulation on Hadoop 1.x Ubuntu 14.04 using java implemented on Core i7 (9th generation) CPU of a Lenovo laptop with 16GB RAM and 64-bits Windows 10 operating system. Moreover, we have used files created randomly.

Figure 9 shows the time taken to implement the encryption of the recent proposed Algaradi and Rama [6], and standard blowfish algorithm for files of different sizes, such as 1MB, 10MB, 100MB,200MB, and 500MB.

Our proposed scheme delivers better results that increase its efficiency and performance, where we notice that the results obtained by the proposed to encrypt the data increased the power and safe consumption by about 40% compared to our previous proposed [6], which made it the best. Similarly, the implement took less time of the decryption of the proposed as compared to our previous proposed [6] and the standard blowfish algorithm, where Fig. 10 illustrates that for the same files encrypted previously.

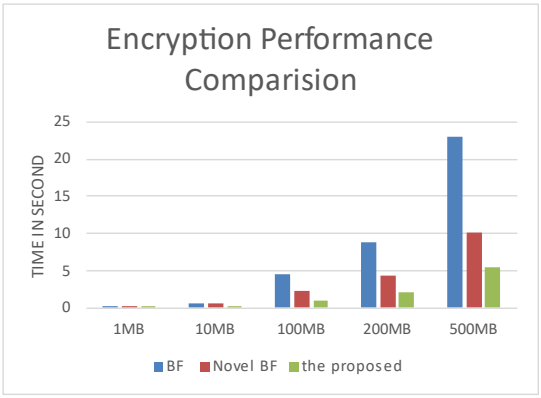

Fig. 9. Encryption performance comparison.

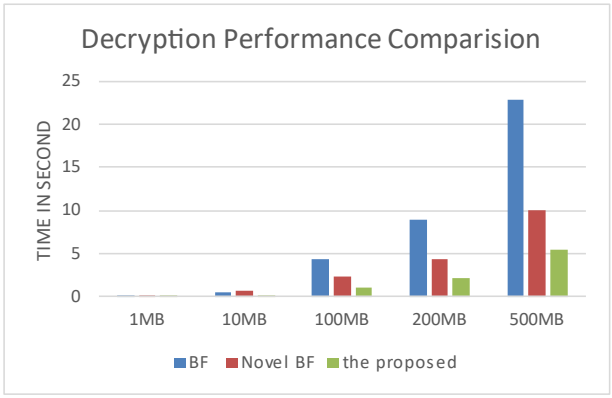

Fig. 10. Decryption performance comparison.

5.1. Performance analysis

The following criteria measure the performances:

- Execution time in terms of encryption and decryption time.
- CPU Process time in the form of throughput.
- Power consumption.

The factors that have been relied upon here to determine and measure the proposed scheme performance is the speed of the algorithm's encrypt/decrypt data blocks of various sizes. The encryption time is known as the time taken of the encryption algorithm to deliver a ciphertext from a plain textual content, where the decryption time is known as the time taken of the decryption algorithm to deliver a plaintext from a cipher textual [29]. It is essential to mention that we used the encryption execution time results to calculate the encryption scheme throughput, where the throughput of the encryption scheme is the total size of plaintext in bytes divided by the total encryption execution time.

$$Throughput = \frac{Total\ size\ of\ Original\ Text(Kbytes)}{Total\ Execution\ Time} \quad (18)$$

It is required to consider data files of different sizes (1MB to 500 MB) to calculate the above equation, where the algorithms evaluated in terms of the time required to encrypt and decrypt the data files.

Table 4 shows the time needed for the encryption and decryption process for the normal blowfish, Algaradi and Rama [6], and the proposed scheme. As well, it shows the average time for different sizes of input files. Finally, compare the throughput for each algorithm. As we observe through visible results, the proposed scheme performs better compared with others regarding time consumption, throughput. It gives the highest degree of throughput, which means the proposed technique's power consumption is the least.

Table 4. Comparison results of the performance analysis of the Encryption/Decryption algorithms.

	Encryption Time in ms			Decryption Time in ms		
Input file in kb	**Blowfish**	**Novel blowfish [6]**	**Proposed**	**Blowfish**	**Novel blowfish [5]**	**Proposed**
1015	113	155	81	127	152	62
10150	461	615	191	498	661	153
102920	4526	2307	1080	4385	2266	1061
205839	8898	4343	2182	9281	4454	2016
532876	22919	10079	5382	23900	10088	4888
Average Time	7383.4	3499.8	1783.2	7.638.2	3524.2	1636
Through put	23.1004	48.7342	95.6482	22.3298	48.3967	104.2542

Figures 11 and 12 clearly show the throughput results presented in Table 4. in encryption, and decryption, respectively.

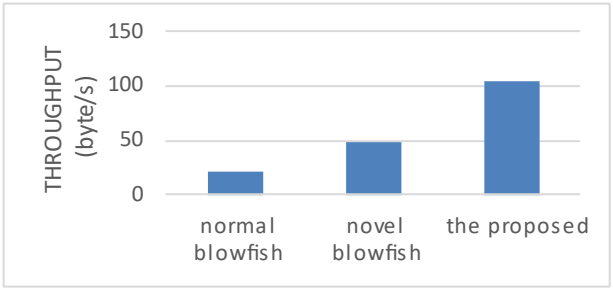

Fig. 11. Throughput of encryption algorithms.

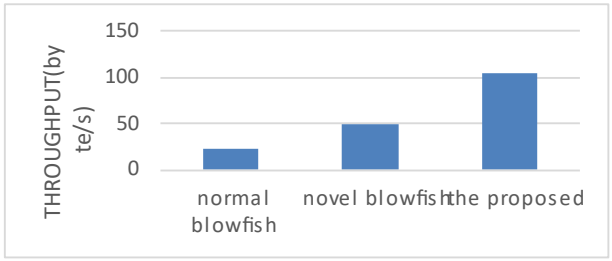

Fig. 12. Throughput of decryption algorithms.

5.2. Features comparison

Computational speed, Throughput encryption/decryption, Power consumption, and Memory usage are essential critical roles in data security that are considered a concerning issue. Here, we used the Blowfish algorithm, a novel blowfish [6], and the proposed scheme for overall performance evaluation.

Table 5 presents the capability comparison of the standard blowfish algorithm, the novel blowfish [6], and the proposed scheme, where it illustrates that the novel blowfish algorithm is more secure than the standard blowfish algorithm. However, the proposed scheme is more secure than both of them. It became concluded that the proposed scheme is to get higher information security and superior performance than others in terms of throughput encryption, throughput decryption, power consumption, memory usage, and confidentiality.

Table 5. The capability comparison of algorithms.

Features	Blowfish	Novel Blowfish [6]	The proposed
Computational Speed	Moderate	Fast	Very Fast
Throughput Encryption	Low	Medium	High
Throughput Decryption	Low	Medium	Very High
Power Consumption	High	Medium	Lowest
Memory Usage	Medium	High	Low
Confidentiality	Medium	High	Very High

Figure 13 shows the difference between the results obtained using MapReduce and without MapReduce. As we see, the proposed with MapReduce gave us less time to consume. Moreover, the proposed scheme was the best and the least.

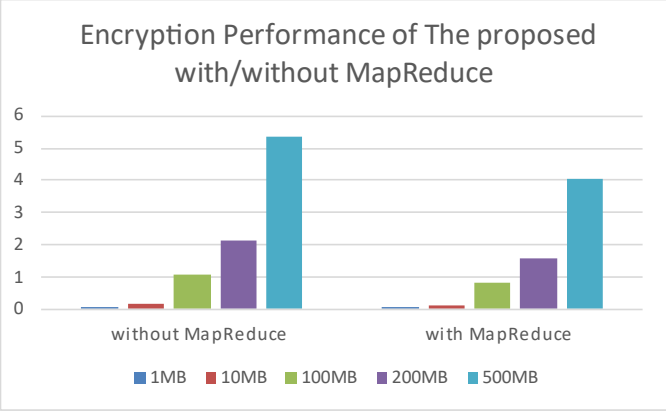

Fig. 13. Encryption performance comparison with and without MapReduce.

6. Conclusions

In this work, we proposed a new scheme for encryption performance improvement in a big data environment using MapReduce as an attempt to avoid data security problems and help solve them at the HDFS storage level. The proposed achieved very satisfactory results concerning the encryption/decryption timing graphs, especially if

it is used in the Hadoop cluster using MapReduce, as the encryption field, it reduced more the cost than its predecessor that is done without MapReduce as the result of the parallel processing of the map and reduce functions. It meets the purpose, trying to overcome most of the weaknesses that the area faced previously and come up with the best results with ensuring the sensitive data security far from detection for any targeted attempt. The proposed proved its strength by decreasing the power and saving consumption compared to some previous related works, as it achieved success by more than 40%. Moreover, we discussed the most important concepts related to demonstrating the proposed strength, which is usually used to measure the objective evaluation metrics and some of the statistical analysis, Visual Assessment, Security Analysis. It also made positive results regarding throughput, thus balancing security, time-consuming, and performance and surely proved that the proposed is highly qualified to carry out its work and accomplished data security.

References

1. Algaradi, T.S.; and Rama, B. (2019). Static knowledge-based authentication mechanism for hadoop distributed platform using Kerberos. *International Journal on Advance Science, Engineering and Information Technology*, 9(3), 772-780.
2. Adluru, P.; Datla, S.S.; and Zhang, X. (2015). Hadoop eco system for big data security and privacy. *Long Island Systems, Applications and Technology*. Farmingdale, USA, 1-6.
3. Tondon, D.; and Khurana, M. (2017). Security of big data in Hadoop using AES-MR with auditing. *In International Journal of Advanced Research in Computer Science and Software Engineering*, 7(1), 100-105.
4. Al-Rummana, G.; and Shende, G.N. (2018). Homomorphic encryption for big data security a survey. *International Journal of Computer Sciences and Engineering*, 6(10), 503-511.
5. Algaradi, T.S.; and Rama, B. (2018). Big data security: A progress study of current user authentication schemes. *4th International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. Mangalore, India, 68-75.
6. Algaradi, T.S.; and Rama, B. (2019). A novel blowfish based-algorithm to improve encryption performance in Hadoop using MapReduce. *International Journal of Scientific and Technology Research*, 8(11), 2074-2081.
7. Sharma, A.; and Sharma, D. (2016). Big data protection via neural and quantum cryptography. *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India, 3701-3704.
8. Le, D.; Chang, J.; Gou, X.; Zhang, A.; and Lu, C. (2010). Parallel AES algorithm for fast data encryption on GPU. *2nd International Conference on Computer Engineering and Technology*. Chengdu, China, V6-1-V6-6.
9. Kadre, V.; and Chaturvedi, S. (2015). AES–MR: A novel encryption scheme for securing data in HDFS environment using MapReduce. *International Journal of Computer Applications*, 129(12), 12-19.
10. Haldankar, C.; and Kuwelkar, S. (2014). Implementation of AES and blowfish algorithm. *International Journal of Research in Engineering and Technology,* 3(Special: 3), 143-146.

11. Hercigonja, Z. (2016). Comparative analysis of cryptographic algorithms. *International Journal of Digital Technology and Economy*, 1(2), 127-134.
12. De Mauro, A.; Greco, M.; and Grimaldi, M. (2015). What is big data? A consensual definition and a review of key research topics. *4th International Conference on Integrated Information*. Madrid, Spain, 97-104.
13. Desai, S.; Park, Y.; Gao, J.; Chang, S.Y.; and Song, C. (2015). Improving encryption performance using MapReduce. *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*. New York, USA, 1350-1355.
14. Dean, J.; and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM,* 51(1), 107-113.
15. Ghemawat, S.; Gobioff, H.; and Leung, S.T. (2003). The Google file system. *SOSP '03: Proceedings of The Nineteenth ACM Symposium on Operating Systems Principles*. New York, USA, 29-43.
16. Dean, J.; and Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM*, 53(1), 72-77.
17. Mandal, A.K.; Parakash, C.; and Tiwari, A. (2012). Performance evaluation of cryptographic algorithms: DES and AES. *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*. Bhopal, India, 1-5.
18. Manikandan, G.; Krishnan, G.; and Vaidhyanathan, V. (2010). A novel approach to the performance and security enhancement using blowfish algorithm. *International Journal of Advanced Research in Computer Science*, 1(4), 451-454.
19. Kofahi, N.A.; Al-Somani, T.; and Al-Zamil, K. (2003). Performance evaluation of three encryption/decryption algorithms. *2003 46th Midwest Symposium on Circuits and Systems*. Cairo, Egypt, 790-793.
20. Maitri, P.V.; and Verma, A. (2016). Secure file storage in cloud computing using hybrid cryptography algorithm. *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. Chennai, India, 1635-1638.
21. Jayan, A.; and Upadhyay, B.R. (2017). RC4 in Hadoop security using mapreduce. *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*. Chennai, India, 1-5.
22. Jayan, A.; Nair, A.; Upadhyay, B.R.; and Supriya, M. (2016). Performance analysis of modified RC4 cryptographic algorithm using number of cores in parallel execution. *International Journal of Control Theory and Applications (IJCTA)*, 9(21), 225-231.
23. Hodeish, M.E.; and Humbe, V.T. (2018). An optimized halftone visual cryptography scheme using error diffusion. *Multimedia Tools and Applications*, 77(19), 24937-24953.
24. Shivani, S.; and Agarwal, S. (2018). VPVC: verifiable progressive visual cryptography. *Pattern Analysis and Applications*, 21(1), 139-166.
25. Wang, C.; and Shen, H.W. (2011). Information theory in scientific visualization. *Entropy*, 13(1), 254-273.

26. Pierce, R. (2020). Standard deviation and variance. Math is fun. Retrieved July 11, 2021, from http://www.mathsisfun.com/data/standard-deviation.html.
27. Hayes, A. (2021). Coefficient of variation (CV). Retrieved April 16, 2021, from https://www.investopedia.com/terms/c/coefficientofvariation.asp.
28. Wikipedia (2021). Key space (cryptography). Retrieved June 20, 2021, from https://en.wikipedia.org/wiki/Key_space_(cryptography)#cite_note-3.
29. Atia, T.S. (2014). Development of a new algorithm for key and S-box generation in blowfish algorithm. *Journal of Engineering Science and Technology* (*JESTEC*), 9(4), 432-442.