ISSN 2278-3091
Volume 10, No.3, May - June 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse411032021.pdf
https://doi.org/10.30534/ijatcse/2021/411032021

Faster Big Data Encryption Technique Using Key Generation

Galal A. Al-Rummana[1], Abdulrazzaq H. A. Al-Ahdal[2], G. N. Shinde[3]
[1] School of Computational Sciences, S.R.T.M. University, India,Galal300z@gmail.com
[2] School of Computational Sciences, S.R.T.M. University, India,alahdal201211@gmail.com
[3]Yeshwant College, Former Pro-Vice Chancellor, S.R.T.M. University, India, shindegn@yahoo.co.in

ABSTRACT

In the present study, an algorithm for big data encryption has been designed which is concerned with encrypting data in a short time and in a safe manner and difficult to be penetrated by attackers and hackers according to the mechanism used for encryption in this study.

The proposed algorithm is a development of an earlier NSCT algorithm. This development is based on adding a key expansion mechanism to generate keys by F-function, and also the number of cycles have been reduced to get less time. The proposed algorithm has been implemented in three stages; the first stage is implemented by FELICS simulation to find out the efficiency of the algorithm. In the second stage the proposed algorithm is implemented on the Java language to encrypt the text and to find out the execution time for the encryption. In the last step, the proposed algorithm is implemented on the MATLAB language to encrypt images and to know the performance of the algorithm in terms of countering attacks by hackers.

After performing these steps and obtaining the results presented in this study, we can say that the proposed algorithm is capable of dealing with big data and that it can be considered safe and free from risks.

Key words: Big Data, Big Data Security, Big Data Encryption, Cryptography, key Generation.

1. INTRODUCTION

From day to another, information technology is moving at a fast pace towards a very great horizon. The data is increasing tremendously due to the massive use of modern technology that has become present in every part of the world to manage the affairs of people, organizations and governments and facilitate their work. This tremendous acceleration imposes the existence of huge data that are difficult to quantify, for instance, at the home level a large amount of data is generated, so how is the situation at a larger level for companies, institutions and governments. Taking all these in consideration, serious questions should be raised;

- How can we protect this data safely? ,
- Are the traditional protection systems in place now capable of protecting in the same manner as they were previously? And
- Are there no other ways to deal with, analyze and store these data more accurately?

By observing the data, we find that their increase is not limited to an individual level, as we note that there are many areas for data that include science, medicine and other different fields, and this does not include structured data only, but exceeds them to unstructured data and semi-structured data.

When describing these data, the matter is focused on their quantitative size, we can say that they are (terabyte) or (beta byte), and because they are of this enormous size, it is certain that they carry sensitive data, which makes the possibility of data corruption a source of concern and fear for the user. However, it becomes more serious when data corruption happened in government or private institutions. So, it must be protected by strong systems that keep them safe from any attack and damage, and we can also say that volume, variety and velocity are among the most important characteristics of big data [1].

Therefore, in this study, we will talk about solutions that are summarized in improving one of the data encryption techniques used to encrypt big data. This can be done through making a small change in the structure of the previous algorithm, in terms of the key and the number of cycles, as well as a comparison of the new algorithm with some existing algorithms for encrypting big data. This will improve work efficiency and reduce execution time by reducing the number of cycles.

Aim of the study

The aim of this study is to compare the performance and security of the encryption system in the New Stream Cipher Technique for Data Encryption (NSCT), Blowfish, Novel-Blowfish, and the proposed algorithm. The study also aims to calculate the execution time taken in encryption and which one is more appropriate for big data.

2. LITERATURE REVIEW

Generally, the science of cryptography is the study of concealing information and the permission of protection of information from other third parties when communication occurs through an unreliable medium such as the internet. This science has two encryption keys: the public encryption key and the private encryption key.

In big data technology, we deal with symmetric algorithms not asymmetric algorithms because the asymmetric algorithms are very slow and takes very long time. The symmetric algorithms are used in many modern computer systems to provide security for sensitive information.

There are many standard basic algorithms that are used for encryption, but in term of productivity, you may find that the Blowfish algorithm is one of the algorithms that have proven to be effective from unbreakable encryption non-penetration compared to similar algorithms such as DES and T-DES [2], [3]. In the Big data, we need to have the encrypting and decrypting in a shorter time as possible. Returning to the Blowfish, we find that it needs less time than other algorithms such as AES [4].

Thoyazan and B Rama [5] worked to protect large data by improving the Blowfish algorithm and merging it with the Hadoop to ensure a high level of security against any attempted attack. This method is based on the original Blowfish algorithm with a simple modification to process more than one block of 128 bits and reduce the number of cycles. As for the key, it extracts the key for each operation from the predefined main key.

Gupta, et al. [6] improved the security of big data by using the technology of Elliptic Curve Cryptography by storing the data in the cloud. They also introduced in their paper the techniques of processing the big data used to secure the data and they explain that this technology reduces time and money and also leads to secure access to data as well as increases reliability and scalability.

Galal, et al. [7] suggested a protection for large data by using an encryption technique that depends on processing more than one block, dividing each block into two parts, switching between them and merging them, and then applying a logical operation XOR and finally the cycle process. This model shows that there is a large key space that makes the text secure and protected from the attacks and hackers, then it shows the execution time and some performance analysis metrics.

Vinit Gopal Savant [8] discusses Big data security issues and security challenges in a Big data environment and categorizes them into authentication level, data level, network level, and general problem level. He also discusses approaches to encryption techniques.

ShadiAljawarneh, et al. [9] develop and design a resource-efficient encryption algorithm system that applies multi-threaded programming process for encrypting multimedia big data. This proposed system describes a multi-level encryption model that uses the Feistel encrypting system, genetic algorithms, and the Advanced Encryption Standard (AES). They evaluate their system to multimedia data and compared with standard encryption algorithms such as RC6, MARS, 3-DES, DES, and Blowfish for computational runtime and transfer rate for encoding and decoding actions. In addition, a multi-threaded programming approach was adopted to implement the proposed coding scheme in order to enhance the system's efficiency and characteristics.

H.Amellal, et al. [10] studied big data security based on quantum information theory. They discuss the opportunities provided by quantum key distribution (QKD) protocols and quantum algorithms to increase the security of big data. In this regard, the researchers used the BB84 protocol to transport qubits sent via a quantum channel and Grover's algorithm to search NoSql databases in the quantum environment. In order to analyze the effectiveness of quantitative information theory on big data.

3. THE PROPOSED ALGORITHM

The proposed algorithm encrypts the data in a complex manner and by processing the data in blocks, each block used a new key. The goal of improving the algorithm is to improve efficiency and reduce execution time by reducing the number of cycles and also to make key expansion. When designing a new encryption algorithm, it is necessary to focus on the environment in which the algorithm works, the execution time spent, and the results of the encrypted text. This means that the algorithm has enough confusion for the text.

In our proposed algorithm, the file to be encrypted is divided into several blocks, each block has 64 bits, every block is processed individually and includes a sequence of operations such as splitting, switching, logical operations such as xor, then splitting into smaller blocks, and a specific technique for bit selection after that grouping, finally at the end of this sequence there is a 7-cycle rotation process as show in Figure (1).

This process makes the encrypted text difficult to predict and protect it from break-ins, also keeps it safe from attack and hackers. As for what is new in these algorithms is the key generation technology, key expansion, which has been used to generate a key that is difficult to guess because in this technology we use what is called an f-function. We will explain some of the concepts in the following.

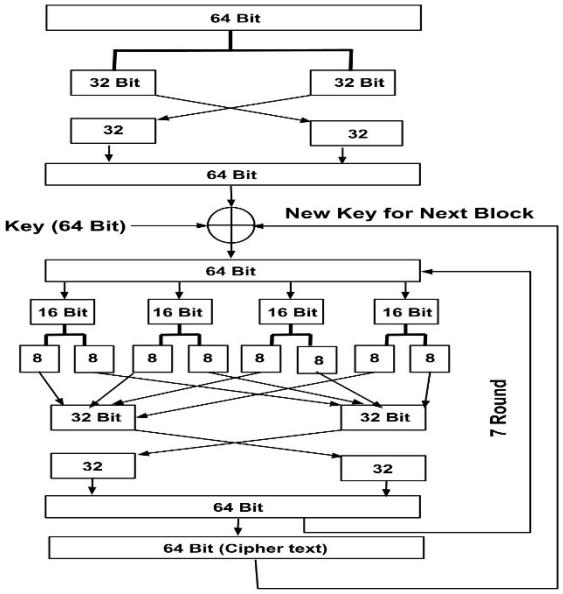

Figure 1: Encryption Process.

3.1 Key Expansion

We all know that the key is the most important part of any encryption algorithm because the person who knows the key will be able to obtain the encrypted text. Therefore, consideration must be given to the importance of the key, the mechanism for generating the key that creates confusion that lead to increase security and difficulty in guessing and predicting. The proposed algorithm uses a 64-bit key to encrypt a 64-bit block. The key is entered by the user, and then the process of generating the new key is done by the method of key expansion, which is shown in Figure (2).

In the key expansion process, the key entered by the user is separated into four sections. Each section contains 16 bits, and then the first bit of each section is grouped into a block and then the second bit, and so on until four blocks are obtained, each block has 16 bits. In the next step, the four blocks will be input on the F-function that will be explained in section 3.2. and the output is four keys each key has 16-bit, now the four keys (GFK1, GFK2, GFK3, GFK4) are combined to obtain the key with 64 bit which the encryption is done by it.

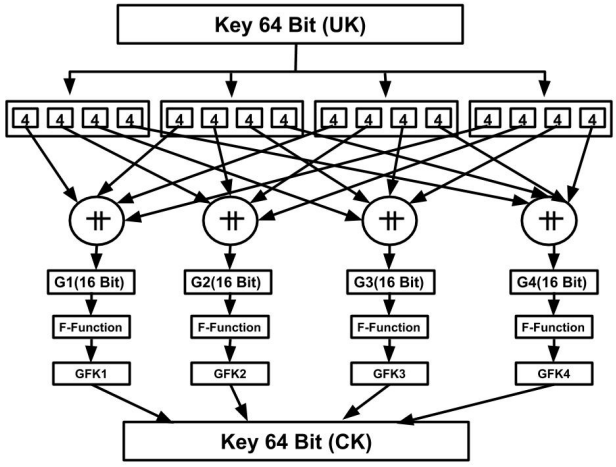

Figure 2: Key Expansion Process.

Kc_i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$P(Kc_i)$	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1
$Q(Kc_i)$	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

Figure 3: P and Q Value.

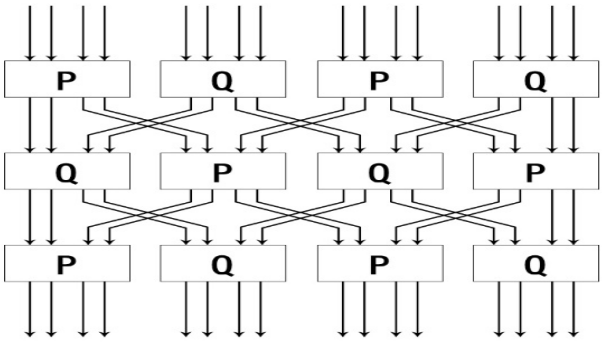

Figure 4: F-Function [13].

3.2 F- Function

This strategy is also called the strategy of confusion and diffusion because it intends to confuse the input values and convert them to other new values, as illustrated in the Figure (3). In Figure (4). We can find an illustration of this strategy and mechanism of this action [11], [12], [13].

3.3 Decryption Process

The decryption process is based on the encryption process but in a reverse order as described in the decryption algorithms in the Algorithms section 3.4.3.

3.4 Algorithms

3.4.1 Key Expansion

Input: 64 bit key i.e. user key(UK).
Output: 64 bit key i.e. cipher key (CK).

1. Divide UK into four group, each group consist of 16 bits.
$$G1 = UK[1]||UK[5]||UK[9]||UK[13]$$
$$G2 = UK[2]||UK[6]||UK[10]||UK[14]$$
$$G3 = UK[3]||UK[7]||UK[11]||UK[15]$$
$$G4 = UK[4]||UK[8]||UK[12]||UK[16]$$
2. Get F-Function for each group G1, G2, G3, G4.
$$GF1 = F - Function\ (G1)$$
$$GF2 = F - Function\ (G2)$$
$$GF3 = F - Function\ (G3)$$
$$GF4 = F - Function\ (G4)$$
3. Generate final Key.
$$CK = GF1||GF2||GF3||GF4$$
4. End

3.4.2 Encryption Procedure

1. Divide 64 bits (plain text) to (L) and (R) both of 32 bit.
2. Swap (L) and (R) and,
3. Merge (L) and $(R)$to the (X).
4. $X = X \oplus CK$

For the first round do the following

- Divide X to four parts each of 16 bits each have (a) and (b)
- Divide each part to two equal array(a) and (b) each of 8 bits
- Recombine $XL = a1 + a2 + a3 + a4$
- Recombine $XR = b1 + b2 + b3 + b4$
- Swap (XL) and (XR)
- Recombine XL and XR
- Repeat step 4 for 7 time

3.4.3 Decryption Procedure

1. Divide 64 bit (cipher text) to (L) and (R) both of 32 bit.

2. Swap (L) and (R) and,
3. Divide each part (L) and (R) into equal array each has 8 bit
4. Recombine each 8 bit into 4 part each of 16 bit as following
   $(L1 + R1), (L2 + R2), (L3 + R3), (L4 + R4)$
5. Recombine 16 part into (X)
6. $X = X\text{^}Ki$
7. Repeat step (1) to step (6) for 7 round
8. Divide (X) to (L) and (R)
9. Swap (L) and (R)
10. Merge (L) and (R) to the (plain text).

4. IMPLEMENTATION

The algorithm was implemented on several parts. The first part was applied to the FELICS simulation and then the code of the algorithm was written and implemented in the Java language and finally the code was written and implemented in the MATLAB language, the implementation is done as follows:

4.1 Implementation and Analysis by FELICS Simulation

In this part, the algorithm was applied to the FELICS simulation to ensure that the steps of the algorithm and the implementation mechanism are correct or not.
This simulation is done by comparing first the entered plain text with the expected plain text, second the input key with the expected key, then the encrypting process begins, finally comparing the cipher text with the expected cipher text [14], [15].

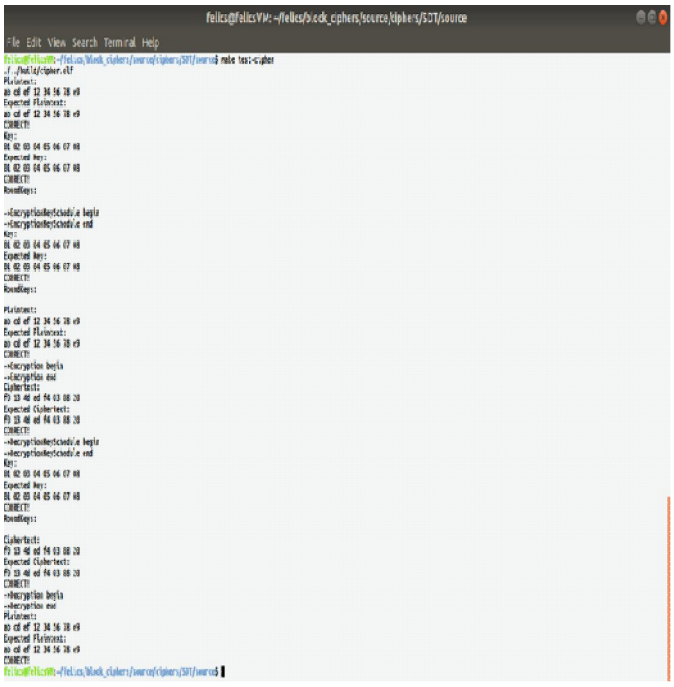

Figure 5: Implementation on FELICS Simulation.

Figure (5) shows the output of the implementation and we note that the algorithm works well, where the text to be encrypted appears identical with the expected text. After that, the appearance of the expected key and key are identical, then appearance of the beginning of the encryption message then the end of the encryption after that the appearance of the cipher text with the expected cipher text, and so on in the decryption process and as shown in Figure 5. As apparently noted through this process there are no errors in the encryption process.

4.2 Implementation and Analysis by Java

In this part, the performance of the algorithm is evaluated and compared with the NSCT algorithm, the Blowfish algorithm, and Novel Blowfish. In this stage the analysis is done only for encrypted text.
This stage was done by using java programming implemented on Core i7 (9th generation) CPU of Lenovo laptop with 16 GB RAM and 64bit Windows 10 operation system, and the analysis was conducted by the following tests.

4.2.1 Security Analysis

From the table 1, regardless of the NSCT algorithm, the convergence of these algorithms becomes evident in many of the characteristics that were compared with them, and this does not mean that there is no difference between them, but rather evidence of the proportionality of the choice in the comparison, the difference will appear in what we will discuss in the following points.

Table 1: Comparison of algorithms

Security algorithm	Cipher type	Key size	Block size	Key space	Round
NSCT	Symmetric block cipher	256-3840	256	2^256 -2^3840	15
Blowfish	Symmetric block cipher	32-448	64	2^32-2^448	16
Novel Blowfish	Symmetric block cipher	64-448	Each operation (128 bits)	2^64-2^448 For each Operation	7
proposed algorithm	Symmetric block cipher	64-448	64	2^64-2^448	7

4.2.2 Key Space

Key space is the most important point that supports the size and durability of the encryption system. It plays an effective role in countering and preventing attacks and brute force attacks. In our proposed algorithm we note that the key space is strong, which is observed during the key generation process by expansion key where a user entered a single block consisting of 64 bits. After that four keys will be generated,

each key consists of 16 bits, the four keys are combined in one block and entered in an XOR operation with the text to be encrypted. The output from the first encryption is entered as a new key for the second block of the text to be encrypted and so on, until we obtain the fully encrypted text. Finally, we can say that the proposed algorithm has a key space starting from 2 ^ 64 and ends at 2 ^ 448, this means that there is no opportunity for brute force attack or hackers to break the key of the proposed algorithm [16].

4.2.3 Execution Time

One of the important points that must be focused on while designing an encryption algorithm concerned with encrypting big data is the time taken to encrypt the data. In Figure (6) an illustration of the time execution taken to encrypt files of gradually different sizes starting from 1 MB to 500 MB.

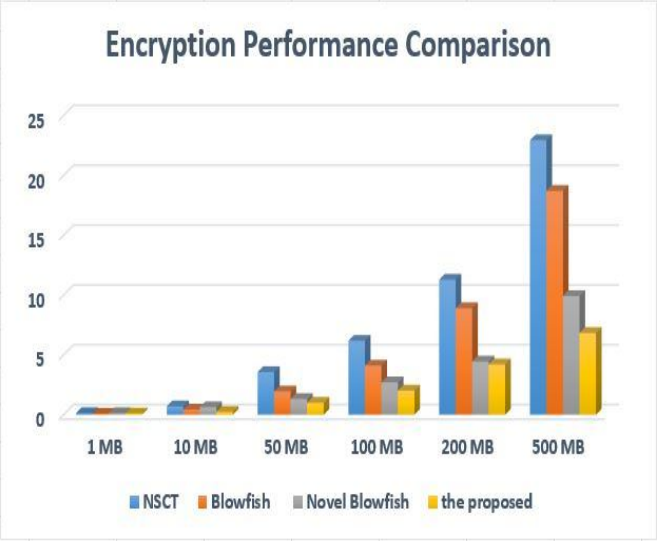

Figure 6: Encryption Performance Comparison.

From figure (6), we note that the proposed algorithm is less time consuming, and this is one of the main reasons that made us emphasize the efficiency of the algorithm. As the results indicated that the time taken in the encryption 1MB file is 0.11 second, 10MB is 0.24 second, 50MB is 1 second, 100MB is 2.02 second, 200MB is 4.21 second, and 500MB is 6.83 second.Finally, we note that the proposed algorithm proves its worth with much more large files than small files. That is, the larger the file size, the faster the processing i.e. suitable with big data.

4.2.4Avalanche Test Effect

This test determines the strength of the encrypted text, where a change of 50 % of the text is an indication that the encryption was good according to the Avalanche test effect standard [17].

$$Avalancheeffect = ((Numberoftheflippingbitsinthecipheredtext / Numberofbitsinthecipheredtext) \times 100)$$

Table 2: Avalanche Test

Variable	Binary code
Key = aaaaaaaaaaaaaaaa	01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001 01100001
Plain text = ABCDEFGHABCDEFGH	01000001 01000010 01000011 01000100 01000101 01000110 01000111 01001000 01000001 01000010 01000011 01000100 01000101 01000110 01000111 01001000
Cipher text = %"#)&'$%"#)&'$	00100101 00100010 00100011 00100000 00101001 00100110 00100111 00100100 00100101 00100010 00100011 00100000 00101001 00100110 00100111 00100100
Bit of change = 44	68.75%
Key = 12345abcdefg6789	00110001 00110010 00110011 00110100 00110101 01100001 01100010 01100011 01100100 01100101 01100110 01100111 00110110 00110111 00111000 00111001
Plain text = 1A2B3C4D5E6F7G8H	00110001 01000001 00110010 01000010 00110011 01000011 00110100 01000100 00110101 01000101 00110110 01000110 00110111 01000111 00111000 01001000
Cipher text = 'rsUuU#——v YqQ	00100111 00000111 01110010 00000101 01110011 01010101 01110101 01010101 00100011 00000011 01110110 00000001 01111111 01011001 01110001 01010001
Bit of change = 42	65.62%
Average	67.18%

From the table (2) from the result obtained here through using Avalanche test, it becomes very clear to us the efficiency of the proposed algorithm with a mean of (67.18) % is very high compared with the minimum result of the Avalanche test

which is > = 50 % to determine the strength of the encrypted text.

4.3 Implementation and Analysis by MATLAB

In this part of the analysis, we will apply the analysis to three randomly selected images from the database [18]. As an essential part of any cryptosystem for images, there must be a difference between the original image and the encrypted image, viz the encrypted image is completely different from the original image.
The figure (7) shows the difference between the original image and the encrypted image. To determine this difference, we must use the following measurements [19, 20, 21].

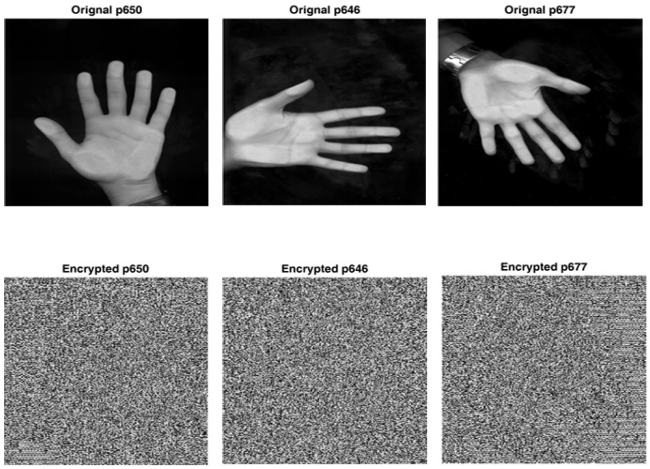

Figure 7: Difference between the original image and the encrypted image.

- **Number of Pixels Change Rate (NPCR)**

It is the percentage of pixels that differs between the original image and the encrypted image, and it is calculated by the following equation:

$$let A is the Original image, A_E Encrypted image,$$
$$MXN size of image, A(i,j) and A_E(i,j)$$
$$the pixels of image at location\ (i,j)$$

$$NPCR = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D(i,j)}{M \times N} \times 100\% \qquad (1)$$

$$\text{With } D(i,j) = \begin{cases} 0 & if A(i,j) = A_E(i,j) \\ 1 & otherwise \end{cases},$$

If the result of this difference is greater than 99%, then this means that the proposed algorithm is strong and from the table (3) we notice that the values are greater than 99%, which indicates that the pixel locations changed randomly.

- **Unified Average Changing Intensity (UACI)**

It is an expression of the average intensity of the difference between the original and the encrypted image, and it is calculated by the following equation:

$$let A is the Original image, A_E Encrypted image,$$
$$MXN size of image, A(i,j) and A_E(i,j)$$
$$the pixels of image at location\ (i,j)$$

$$UACI = \left[\sum_{i=1}^{M}\sum_{j=1}^{N} \frac{|A(i,j) - A_E(i,j)|}{255}\right] \times \frac{100\%}{M \times N} \qquad (2)$$

If the result of this difference is around 33%, then this means that the proposed algorithm is strong and from the table (3) we note that the values are greater than 33%, which indicates that it is greater than the acceptable range.

Table 3: NPCR and UACI Value

Image	NPCR	UACI
p650_l_s1	99.57	36.16
p646_l_s2	99.65	39.56
p677_l_s3	99.63	37.61

4.3.1 Key Sensitive

The proposed algorithm should be of high sensitivity to the keys, which means that a simple modification of only one bit leads to a completely different image. The algorithm was tested by encrypting images by using the K1 and decrypting by using K1 as well as K2, the different between K1 and K2 only one bit.

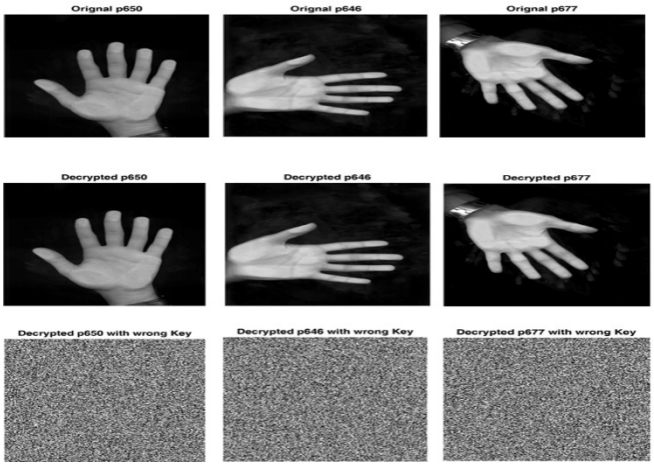

Figure 8: Different Decrypt Image using K1 and K2.

We observed when we decrypt the image by using K1 will get the original image and when we use K2 will get different image as shown in figure (8). Regarding to these results we can say the proposed algorithm is robust against exhaustive attack.

4.3.2 Statistical Analysis

In this part of the test, we will focus on a statistical analysis in order to prove the safety of the proposed algorithm against statistical attacks. Shannon [22], proposed two methods based on confusion and diffusion in order to counteract powerful attacks based on a statistical analysis.

- **Histogram analysis**

In this part, the graph of the encrypted image should be uniform histogram. Figure (9) shows that the pixels of the encrypted image are uniform distributed. This means that the algorithm possesses high security against statistical attacks.

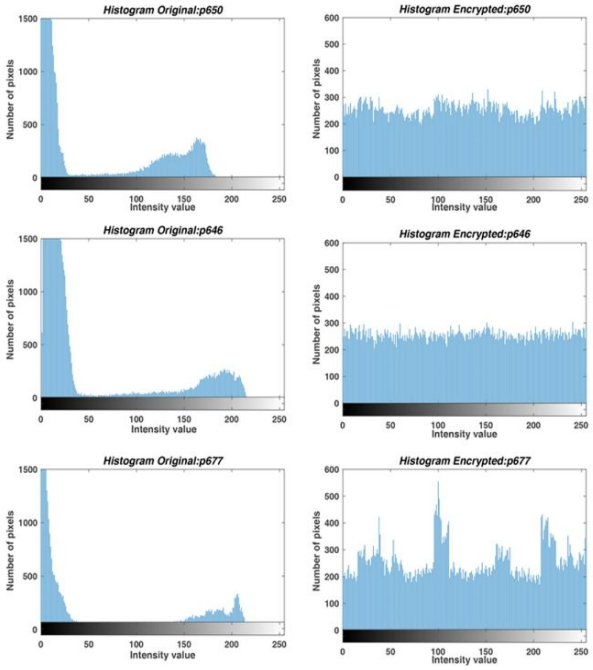

Figure 9: Histogram of Original and encrypted Images.

- **Correlation analysis**

It is used to calculate the correlation relationship between the encrypted data and the data before encryption. It is also characterized by its ability to cover the characteristics of the image [23].
In this part. We note that the correlation coefficient of pixels adjacent to the encrypted image are very small, closer to zero that is an indication of the strength of the proposed algorithm according to the following mathematical expression:

$$CorrCoef = \frac{cov(x,y)}{\sqrt{Var}(x) \times \sqrt{Var(y)}} \quad (3)$$

$$Var(x) = \frac{1}{N}\sum_{i=1}^{N}[(x_i - E(x))^2] \quad (4)$$

$$Cov = \frac{1}{N}[(x_i - E(x)) \times (y_i - E(y))] \quad (5)$$

Where, the correlation coefficient is CorrCoef and the covariance of Cov(x,y) is pixel x and y. Var(x) is the pixel value variance in an image, E(x) is the value operator predicted and N is the total number of pixels in the matrix.

Table 4: Correlation of Image.

Image	Correlation Before Encryption	Correlation After Encryption
p650_l_s1	0.9958	0.0531
p646_l_s2	0.9886	0.0181
p677_l_s3	0.9924	0.1054

- **Information entropy analysis**

This analysis is applied to image to measure randomness in the image and is considered one of the important analysis to prove the security of the encryption. If the result of entropy is high, then this is an evidence of the safety of the encryption algorithm [16]. We notice from the table (5) that the proposed algorithm has a higher output, which makes the security of the encryption high and is mathematically represented as follows:

$$Entropy(H) = \sum_{i=0}^{255} P(x_i)\log_2 P(x_i) \quad (4)$$

Table 5: Entropy Analysis.

Image	Entropy Before Encryption	Entropy After Encryption
p650_l_s1	5.6988	7.9924
p646_l_s2	6.2106	7.9967
p677_l_s3	5.1785	7.9578

5. CONCLUSION

In this paper, we developed a previous algorithm called NSCT. This development is based on the addition of key expansion, which works with F-function to generate keys.
We divide the text to be encrypted into several blocks, each block contains 64 bits. The key generated by the key expansion process is entered and combined with this block by a logical operation called XOR to obtain a new block that contains 64 bits. A sequence of operations is performed on this block to obtain Encrypted text.
What is significant in this algorithm is that each cycle is encrypted with a new key based on the result of the previous cycle and completely different from the previous key.
We calculated the execution time taken for encryption and compared it with the algorithms that operate on the same mechanism. It has been concluded that the proposed algorithm consumes less time.
In addition to that, we calculated the key space of the algorithm to ensure that the algorithm is resistant to attacks such as brute force attack.
Using avalanche test to test the algorithm it has been found that the result was 67.18% which proves the efficiency of the encrypted text.
To test the resistance of the exhaustive attack of the algorithm the key sensitive was also used finally we tested the algorithm using statistical analysis to make sure that the algorithm is against statistical attacks.
From the results obtained, we can say that the proposed algorithm is able to deal with big data effectively and safely.

REFERENCES

1. G. A. AL-Rummana and G. N. Shende, "**Homomorphic Encryption for Big Data Security A Survey**," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 503–511, 2018. DOI: 10.26438/ijcse/v6i10.503511.
2. V. Vaidhyanathan and G. Manikandan, "**A Novel Approach to the Performance and Security Enhancement Using Blowfish Algorithm**," pp. 451–454, 2010.
3. N. A. Kofahi, T. Al-Somani, and K. Al-Zamil, "**Performance evaluation of three encryption/decryption algoriithms**," no. June 2018, pp. 790–793, 2006. DOI: 10.1109/mwscas.2003.1562405.
4. P. V. Maitri and A. Verma, "**Secure file storage in cloud computing using hybrid cryptography algorithm**," *Proc. 2016 IEEE Int. Conf. Wirel. Commun. Signal Process. Networking*,WiSPNET 2016, pp. 1635–1638, 2016. DOI: 10.1109/WiSPNET.2016.7566416.
5. T. S. Algaradi and B. Rama, "**A Novel Blowfish Based-Algorithm To Improve Encryption Performance In Hadoop Using Mapreduce**," *Int. J. Sci. Technol. Res.*, vol. 8, no. 11, pp. 2074–2081, 2019.
6. S. Gupta, S. Vashisht, D. Singh, and P. Kushwaha, "**Enhancing Big Data Security using Elliptic Curve Cryptography**," 2019 *Int. Conf. Autom. Comput. Technol. Manag. ICACTM 2019*, pp. 348–351, 2019. DOI: 10.1109/ICACTM.2019.8776764.
7. G. A. AL-Rummana, G. N. Shinde, and A. H. A. Al-Ahdal, "**MapReduced Based: A New Stream Cipher Technique for Data Encryption**," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 5, pp. 763–769, Jun. 2020. DOI: 10.35940/ijeat.E9394.069520.
8. V. G. Savant, "**Approaches to Solve Big Data Security Issues**," vol. 3, no. 3, pp. 425–428, 2015.
9. S. Aljawarneh, M. B. Yassein, and W. A. Talafha, "**A resource-efficient encryption algorithm for multimedia big data**," Multimed. Tools Appl., vol. 76, no. 21, pp. 22703–22724, 2017. DOI: 10.1007/s11042-016-4333-y.
10. H. Amellal, A. Meslouhi, and A. El Allati, "**Secure Big Data using QKD protocols**," *Procedia Comput. Sci.*, vol. 148, pp. 21–29, 2019. DOI: 10.1016/j.procs.2019.01.003.
11. P. Barreto and V. Rijmen, "**The Khazad Legacy-Level Block Cipher**," *NESSIE Work.*, no. January 2000, p. 15 pages, 2000.
12. A. H. A. Al-ahdal, G. A. Al-rummana, G. N. Shinde, and N. K. Deshmukh, "*International Journal of Computer Sciences and Engineering* Open Access **NLBSIT : A New Lightweight Block Cipher Design for Securing Data in IoT Devices**," vol. 8, no. 10, 2020.
13. A. H. A. Al-Ahdal, G. A. AL-Rummana, G. N. Shinde, and K. D. Nilesh, "**Security Analysis of a Robust Lightweight Algorithm for Securing Data in Internet of Things Networks**," pp. 1–12, 2020.
14. D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. L. Corre, and L. Perrin, "**Felics–fair evaluation of lightweight cryptographic systems**," *NIST Work.* Light. Cryptogr., vol. 128.
15. https://www.cryptolux.org/index.php/FELICS.
16. M. E. Hodeish, L. Bukauskas, and V. T. Humbe, "**A new efficient TKHC-based image sharing scheme over unsecured channel**," *J. King Saud Univ. - Comput. Inf. Sci.*, Aug. 2019. DOI: 10.1016/j.jksuci.2019.08.004.
17. C. Echeverri, "**Visualization of the Avalanche Effect in CT2**," vol. 2016, 2016.
18. A. Magalh˜ et al., "**Hand Geometric Points Detection Competition Database**," 2011. *Available: httpwww.fe.up.pt~hgc2011*.
19. K. Loukhaoukha, M. Nabti, and K. Zebbiche, "**An efficient image encryption algorithm based on blocks permutation and Rubik's cube principle for iris images**," 2013 *8th Int. Work. Syst. Signal Process. Their Appl. WoSSPA*2013, no. October, pp. 267–272, 2013. DOI: 10.1109/WoSSPA.2013.6602374.
20. G. Chen, Y. Mao, and C. K. Chui, "**A symmetric image encryption scheme based on 3D chaotic cat maps**," Chaos, Solitons and Fractals, vol. 21, no. 3, pp. 749–761, 2004. DOI: 10.1016/j.chaos.2003.12.022
21. M. E. Hodeish and V. T. Humbe, "**An Optimized Halftone Visual Cryptography Scheme Using Error Diffusion**," *Multimed. Tools Appl.*, vol. 77, no. 19, pp. 24937–24953, 2018. DOI: 10.1007/s11042-018-5724-z.
22. C. E. Shannon, "**Communication theory of secrecy systems**. 1945.," *MD. Comput.*, vol. 15, no. 1, pp. 57–64, 1998.
23. B. S. Al-Attab, H. S. Fadewar, and M. E. Hodeish, "**Lightweight Effective Encryption Algorithm for Securing Data in Cloud Computing**," *Comput. Commun. Signal Process.*, vol. 810, pp. 105–121, 2019. DOI :https://doi.org/10.1007/978-981-13-1513-8_13.