LAUTECH Journal of Engineering and Technology 18 (1) 2024: 87-97

INVESTIGATING THE EFFECT OF DOUBLE EXCLUSIVE OR OPERATION IN TRIPLE DATA ENCRYPTION ALGORITHM

[1*]Aru O. E., [2]Udo E. U., [3]Chiagunye, T. T. and [4]Promise O. N.

[1,3,4] *Department of Computer Engineering, Michael Okpara University of Agriculture, Umudike, Abia Sate, Nigeria.*
[2] *Department of Electrical/Electronic Engineering, Michael Okpara University of Agriculture, Umudike, Abia Sate, Nigeria.*

Corresponding author's email address: aru.eze@mouau.edu.ng; Phone number: 08062179049.

ABSTRACT

This paper investigates the effect of double-exclusive OR operation in triple data encryption algorithms. There is an increase in hacker threat problems on web servers, despite the existing Cyber security techniques on web servers. Other researchers study the challenges and issues in web security without any provision of a system that will mitigate the hackers' actions in solving the identified web security problems. A robust web server must be built to protect sensitive and confidential data from unauthorized users by building an encryption standard. The method employed involves the design of triple data encryption algorithms (TDEA) with double exclusive OR operation that uses three and two keys. These keys operate on data thrice more than data encryption does. The results obtained reveal that the triple data encryption algorithm with double exclusive OR (DXOR) execution speed was faster than the triple data encryption algorithm without DXOR and it also consumes more memory to initialize the subkeys than the TDEA without DXOR. Also, it is observed from the results that for a file size of 1 MB and 15MB the average response time for data encryption algorithm (DEA) are 0.15 and 2.13 seconds respectively. In conclusion, the effect of double exclusive OR operation in existing triple data encryption algorithms was investigated to improve the security and avalanche features of the conventional data encryption standards.

Keywords: *Double exclusive OR, Encryption, Symmetric encryption algorithm, Triple data encryption algorithm, Web server*

INTRODUCTION

Despite numerous existing data security solutions, the problem of digital theft and electronic hacking has led to apathy in the usage of internet technology. This is because the delicacies of human relationships, such as reciprocal trust in electronic relationships are not usually recognized. Currently, data security and electronic media are used for the majority of communication (Devanshu, 2018).

Data management in computer networks and communication technologies is urgently needed and public communication networks must be used to exchange information. Millions of people create and exchange massive amounts of information every day via the internet in a variety of disciplines including banking services, legal and financial documents and medical reports (Tiago and Carlos, 2016).

The data encryption algorithm uses substitution and transposition techniques to repeat a simple operation. Private key encryption is the algorithm that uses just one key for both encryption and decryption. The lifetime of the key in conventional cryptography used for data encryption is equal to the lifetime of the transaction. The data encryption standard (DES) key is an 8-bit string that has a 7-bit key and a parity bit on each bit (Arunima *et al.*, 2014). The DES algorithm divides the primary text into 64-bit blocks during cryptography. This algorithm encrypts characters one at a time using a block that is divided in half. Under key control, characters are permuted 16 times to create an encrypted 64-bit text (Hamdan *et al*., 2010).

Due to its considerable key length which is longer than most key lengths associated with other encryption modes, the triple data encryption algorithm is favorable. However, the National Institute of Standards and Technology (NIST) replaced the DES algorithm with the advanced encryption standard (AES). Three separate DES are used on the same plain text in the triple data encryption standard. This is how the selection strategy works and it makes use of three different kinds of keys. However, all previously used keys are unique while two keys are identical, one key is unique and three keys are identical (Akshitha *et al*., 2020).

The triple data encryption standard, which is derived from a single DES, encrypts input data three times and uses three sub-keys and key padding. The keys must be extended to 64 bits in length. This technology is covered under the ANSIX9.52 standards (Marwa *et al*., 2016). However, to provide good data security and privacy, an additional XOR operation is used in this paper to perform an exclusive logical operation between the left plain text and the right plain text. The triple data encryption standard (Triple-DES) algorithm makes use of the data encryption standard block three times to increase the key size to 192 bits while strengthening the encryption and decryption strength of DES. By obtaining the keys the dependability of the data can be improved over time (Mohamed and Yousef, 2014).

The DES algorithm structure is based on the Feistel function that divides the block into two halves. The function (f) was based on the four stages such as expansion, key mixing, substitution and permutation. The number of rounds applied for the DES is 16 and was used for the processing to encrypt the message. 64 bits that are a function of the input message and the key make up the output after 16 rounds (Kong, 2016). DES is mostly used for information sharing in the financial sector and the military. This is because the DES uses a 56-bit key and security is a big concern because cryptanalysts are attempting to decipher an encrypted message through key exhaustion (Albrecht *et al.,* 2016).

Kaur and Kumari (2014) carried out a research study to establish database encryption web applications using the transposition substitution folding shifting (TSFS) algorithm. This encryption algorithm was a systematic database encryption algorithm that was enlarged to provide high security. The research experiment revealed that the enhanced TSFS encryption algorithm performs data encryption standard algorithm and advanced standard encryption algorithm in terms of query executed time and database added size. The shortcoming was that the TSFS encryption was not evaluated in terms of the data transmission on the virtual network to determine its strength in protecting data from hackers.

According to Prapulla *et al.* (2017), research was conducted on mobile agent data security that employed triple data encryption standards. The research work was mainly concentrated on an agent-based network without the implementation of web-based applications. Thus Smitha, (2018) employed cloud computing research for future-generation technology where users can keep their data based on the requirements. The author made use of advanced encryption standards as a symmetric cryptographic algorithm to handle cloud data insecurity. The results indicated that the proposed triple data encryption algorithm has performed advanced encryption standards in cloud computing and data security. There is an increase in hacker threats on the web server. The objective is to investigate the effect of double exclusive OR operation to strengthen the performance of data security in the triple data encryption algorithm.

MATERIALS AND METHOD

The materials used in this paper are a window-based system with 32GB of memory and an Intel i74770 3.4GHz CPU, Visual Studio IDE for coding, MATLAB/Simulink, IDE application for simulation and validation, developer C++ and a virtual web server for system demonstration. The method used involves the design of a triple data encryption algorithm generated from simulation with double exclusive OR operation using three and two keys. The triple data encryption algorithm with double exclusive OR operation with three keys implements thrice what the data encryption algorithm normally implements once with three different sets of keys, namely K1, K2 and K3 at each DEA process. Again, it performs the data encryption algorithm on the original text using K1 to get the encrypted text and also performs the data encryption algorithm text with a different key K2. Finally, it performs the data encryption algorithm on the second cipher text to get the third cipher text at each of the three DEA operations and the operation of the exclusive OR was performed twice.

However, the triple data encryption algorithm with double exclusive OR operation with two keys does thrice what the data encryption algorithm normally does once. Also, it encrypts the plain text with key K1 and the next step involves a decryption of the output cipher text with key K2 to obtain the output of the second cipher text. Again, the output of the second cipher text was encrypted using key K1 to form the output of the third cipher text and key K1 was used to perform DEA twice on each of the original plain text and the second cipher text.

The implementation of the double exclusive operation at each round in the data encryption algorithm involves two processes. The first XOR operation occurs between the initial left plain text (LPT) and the initial right plain text (RPT) to form the new left plain text and the new right plain text. The second XOR operation occurs between the output of a permutation operation leading to a 48-bit right plain text and the initial left plain text to form the new right plain text at round 1. The second XOR operation occurs between the results of the P-box permutation and the initial left plain text before switching. Figure 1 shows the flowchart of a double exclusive OR operation at each of the 16 rounds of a data encryption algorithm.

RESULTS AND DISCUSSION

The developed double XOR algorithm was implemented in the data encryption algorithm scheme and the results were compared with the triple encryption algorithm with 3 and 2 keys using file sizes ranging from 1 MB to 15 MB. For security evaluation, the avalanche effect security metric was compared. The performance evaluation of the data encryption algorithm and the number of times the process was applied was based on execution speed using different memory sizes.

In this paper, the memory size is from 100 MB to 1200 MB. Data encryption and the triple data encryption algorithm with and without a double XOR operation were all checked for execution speed. The encryption follows a series of compression, division, permutation and combination processes. At the end of the initial permutation, the expanded right plain text is exclusive ORed with each available key for each of the 16 rounds. Figure 2 shows the double exclusive OR data encryption standard algorithm.

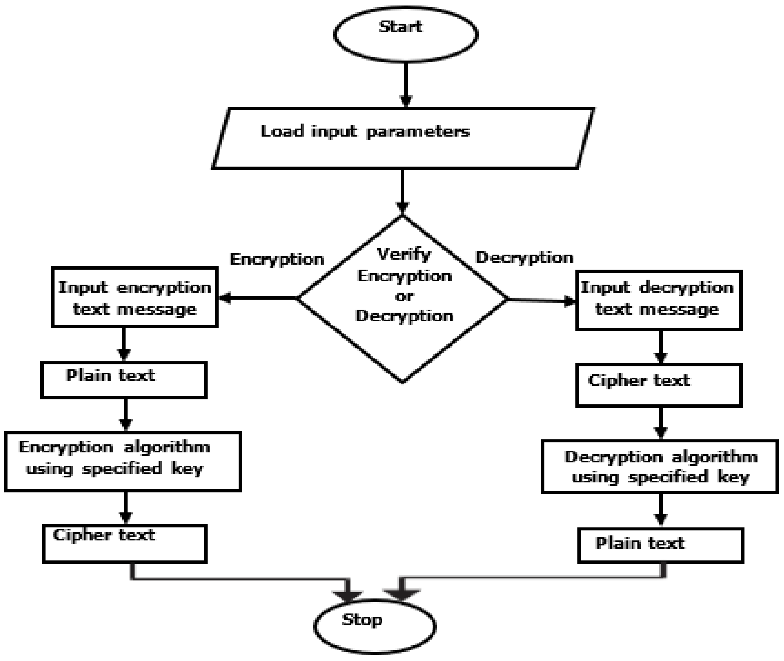
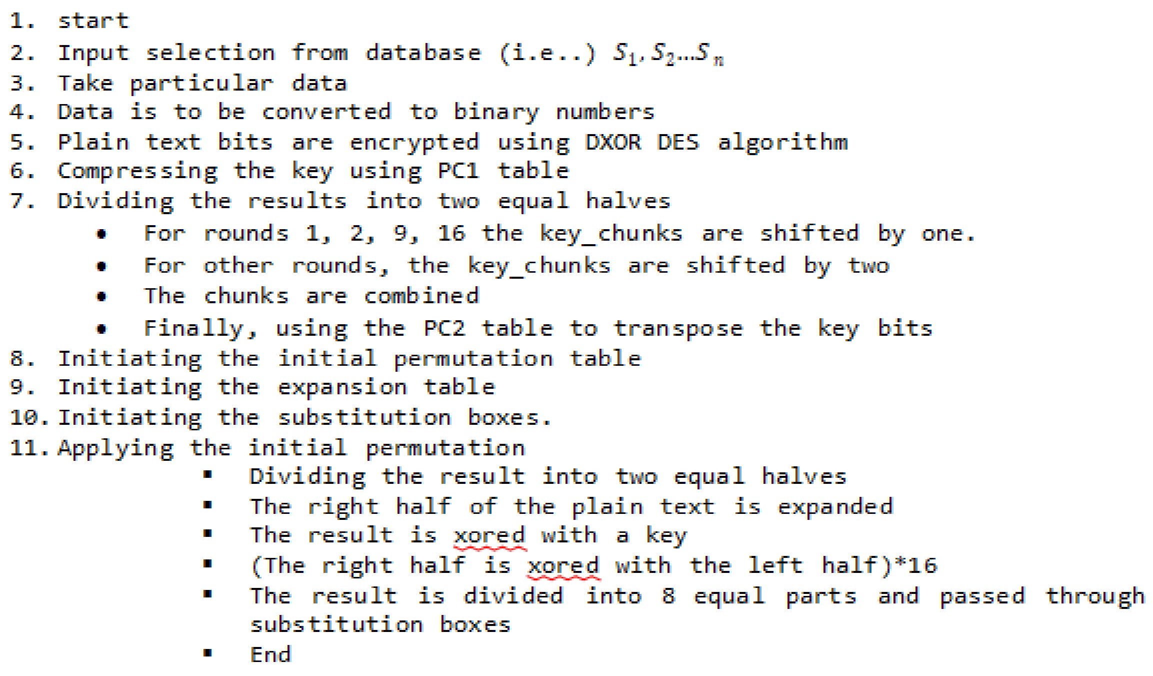

Start
Load input parameters
Verify Encryption or Decryption
Encryption
Decryption
Input encryption text message
Plain text
Encryption algorithm using specified key
Cipher text
Input decryption text message
Cipher text
Decryption algorithm using specified key
Plain text
Stop

Figure 1: Flow chart of the data encryption and decryption in the data encryption algorithm

Table 1: Results obtained from the Double Exclusive OR operation

LEFT PLAIN TEXT	RIGHT PLAIN TEXT	NEW LEFT PLAIN TEXT	NEW RIGHT PLAIN TEXT	LEFT PLAIN TEXT	RIGHT PLAIN TEXT	NEW LEFT PLAIN TEXT	NEW RIGHT PLAIN TEXT
		ROUND 1				**ROUND 1**	
0	1	1	1	0	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	0	0
1	0	1	0	0	1	1	0
1	1	0	1	0	1	1	0
0	0	0	1	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	1
0	0	0	1	1	0	1	0
1	0	1	0	1	1	0	1
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	0	0	0	0	1	1	1
1	1	0	1	0	0	0	1
1	0	1	1	1	0	1	1



```
1. start
2. Input selection from database (i.e..) $S_1, S_2 … S_n$
3. Take particular data
4. Data is to be converted to binary numbers
5. Plain text bits are encrypted using DXOR DES algorithm
6. Compressing the key using PC1 table
7. Dividing the results into two equal halves
      • For rounds 1, 2, 9, 16 the key_chunks are shifted by one.
      • For other rounds, the key_chunks are shifted by two
      • The chunks are combined
      • Finally, using the PC2 table to transpose the key bits
8. Initiating the initial permutation table
9. Initiating the expansion table
10. Initiating the substitution boxes.
11. Applying the initial permutation
            ▪ Dividing the result into two equal halves
            ▪ The right half of the plain text is expanded
            ▪ The result is xored with a key
            ▪ (The right half is xored with the left half)*16
            ▪ The result is divided into 8 equal parts and passed through
              substitution boxes
            ▪ End
```

Figure 2: Double Exclusive OR Data Encryption Standard Algorithm

Table 2: Memory size and execution time algorithm

	EXECUTION TIME				
MEMORY SIZE	**DEA**	**TDEA (3 KEYS)**	**TDEA (3 KEYS)+DXOR**	**TDEA (2 KEYS)**	**TDEA (2 KEYS)+DXOR**
96	1.5321	1.9883	1.6937	1.7881	1.5321
224	1.5021	1.7331	1.6411	1.7101	1.5320
352	1.4921	1.7213	1.6391	1.7093	1.5221
480	1.5021	1.7330	1.6321	1.7091	1.5111
608	1.4921	1.7213	1.6293	1.7023	1.5101
736	1.4881	1.7119	1.6221	1.7023	1.5221
992	1.4921	1.7213	1.6221	1.7003	1.5101
1120	1.4881	1.7213	1.6201	1.7023	1.5011

In Figure 3, it was observed that during the time taken for encryption, the data encryption algorithm executes faster than the single-stage DEA in the algorithm. Again, for the triple DEA with and without double XOR, it was observed that the implementation of the double XOR operation reduced the execution time of TDEA. Also, during the execution time of TDEA, (2KEYS) + DXOR almost matches the runtime of DEA. The results indicate that TDEA with DXOR executes at a speed faster than TDEA without DXOR. Again, TDEA with DXOR consumes more memory to initialize the subkey and s-box than TDEA without DXOR. Figure 4 indicates the data encryption standard algorithm.

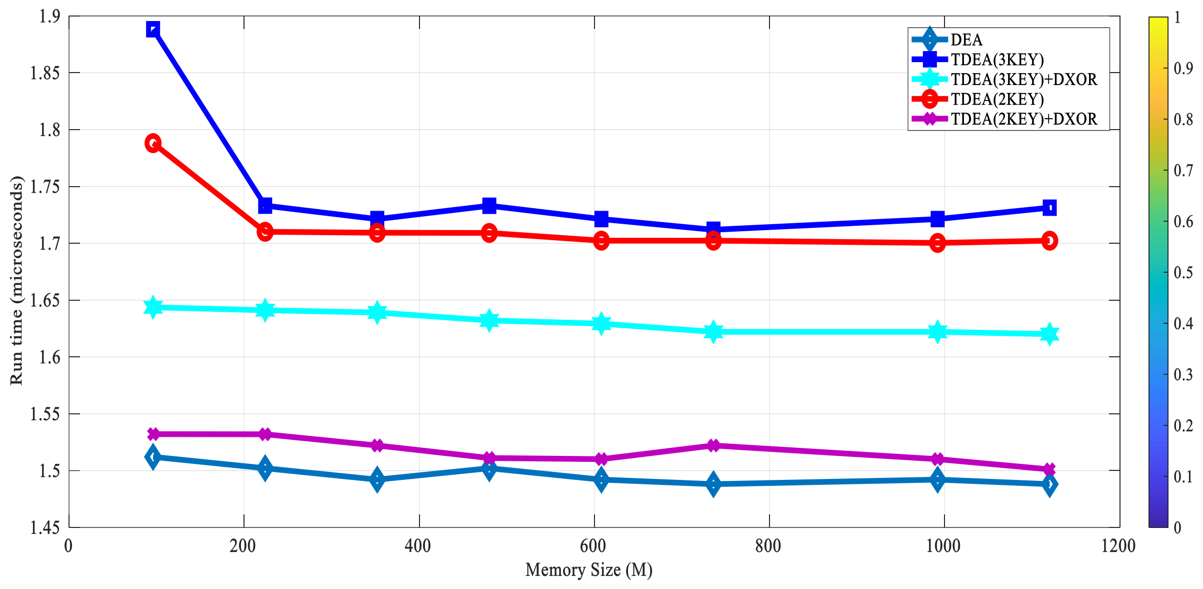
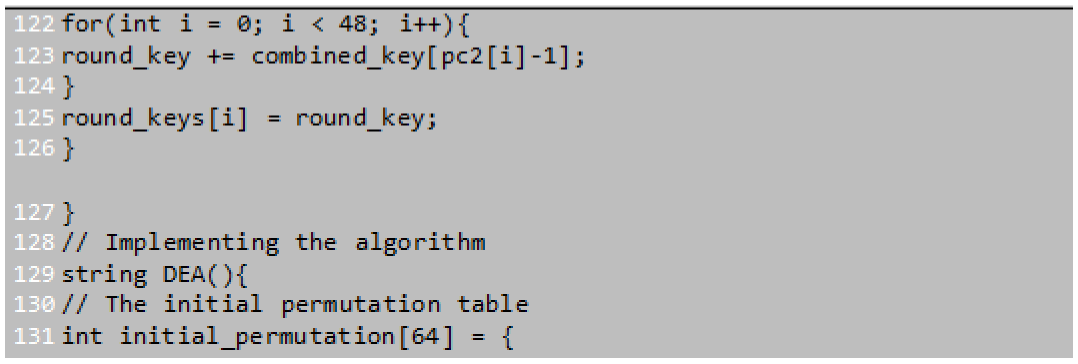


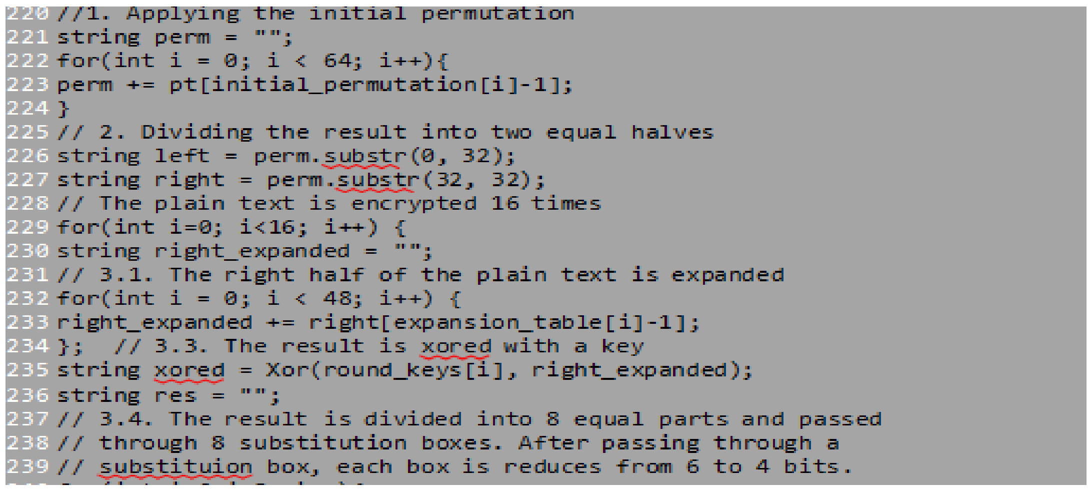
Figure 3: Plot of runtime against memory size



```
for(int i = 0; i < 48; i++){
round_key += combined_key[pc2[i]-1];
}
round_keys[i] = round_key;
}

}
// Implementing the algorithm
string DEA(){
// The initial permutation table
int initial_permutation[64] = {
```

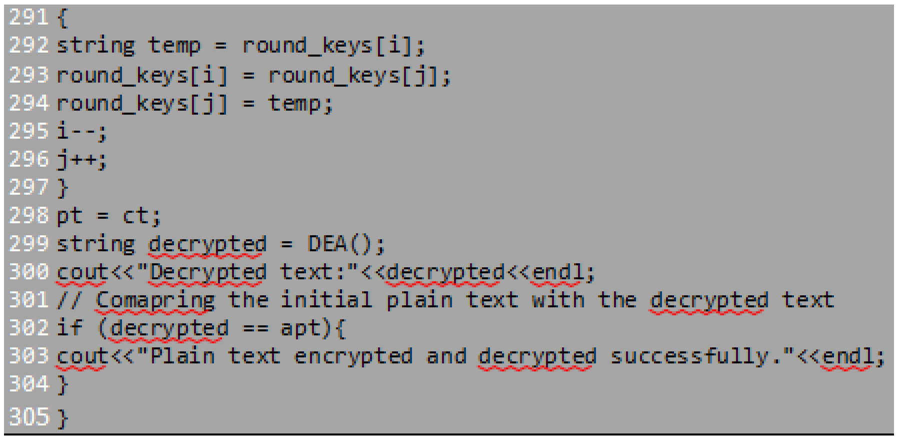
Figure 4: Data encryption standard algorithm



```
//1. Applying the initial permutation
string perm = "";
for(int i = 0; i < 64; i++){
perm += pt[initial_permutation[i]-1];
}
// 2. Dividing the result into two equal halves
string left = perm.substr(0, 32);
string right = perm.substr(32, 32);
// The plain text is encrypted 16 times
for(int i=0; i<16; i++) {
string right_expanded = "";
// 3.1. The right half of the plain text is expanded
for(int i = 0; i < 48; i++) {
right_expanded += right[expansion_table[i]-1];
};  // 3.3. The result is xored with a key
string xored = Xor(round_keys[i], right_expanded);
string res = "";
// 3.4. The result is divided into 8 equal parts and passed
// through 8 substitution boxes. After passing through a
// substituion box, each box is reduces from 6 to 4 bits.
```

Figure 5: Encryption algorithm

Figure 5 shows the encryption algorithm and Figure 6 also shows the decryption algorithm.



```
{
string temp = round_keys[i];
round_keys[i] = round_keys[j];
round_keys[j] = temp;
i--;
j++;
}
pt = ct;
string decrypted = DEA();
cout<<"Decrypted text:"<<decrypted<<endl;
// Comapring the initial plain text with the decrypted text
if (decrypted == apt){
cout<<"Plain text encrypted and decrypted successfully."<<endl;
}
}
```

Figure 6: Decryption algorithm

Table 3: Estimated encryption time of file

File size (KB)	DEA	TDEA (3 KEYS)	TDEA (2 KEYS)	TDEA (3 KEYS)+DXOR	TDEA (2 KEYS)+DXOR
32	0.193	0.6031	0.4521	0.5310	0.410
126	0.372	1.132	1.0310	0.9231	0.710
200	0.413	1.533	1.3774	1.1856	0.9310
246	0.473	1.843	1.632	1.420	1.3471
280	0.610	2.2110	1.8931	1.5210	1.2011
300	0.673	2.530	2.1832	1.600	1.3128

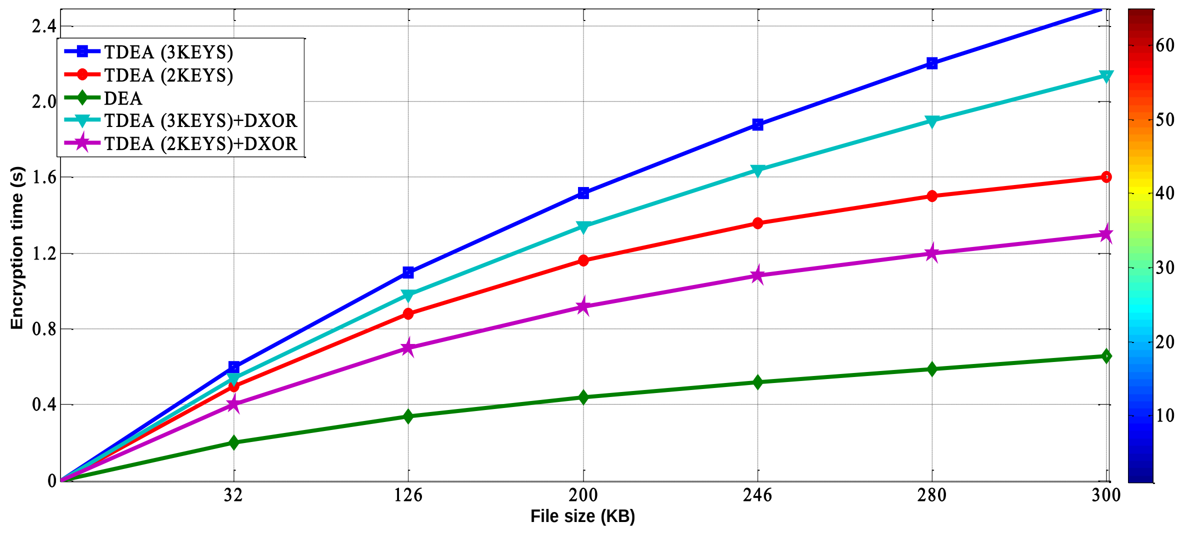
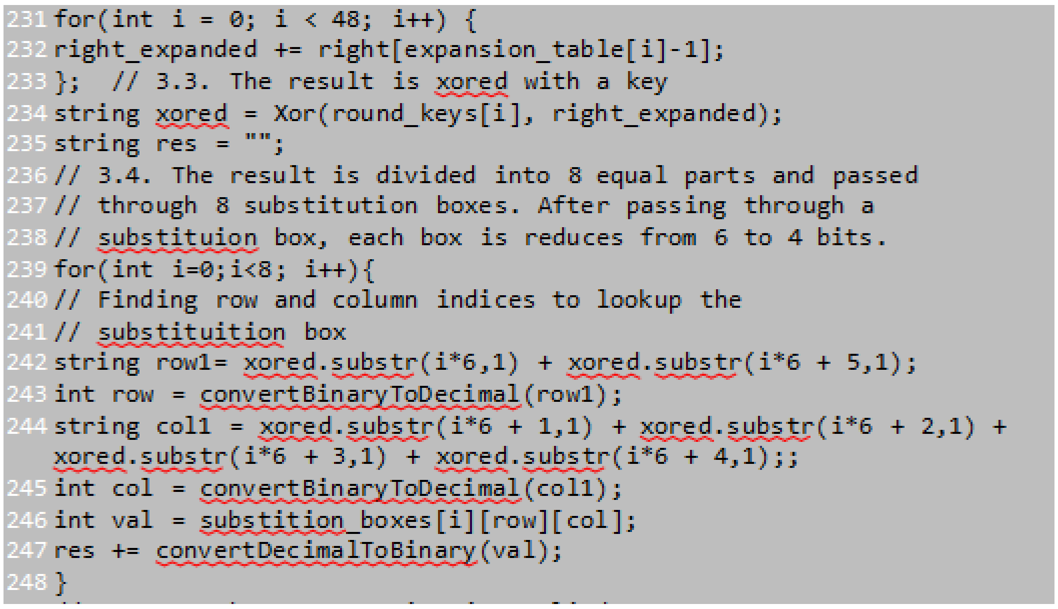


Figure 7: Plot of encryption time against file size for DEA algorithms

The implementation of the double XOR operation shows a higher avalanche effect than the DEA without double XOR operation. The reason is that after the end of the 16 rounds of operation, it provides the generation of a new LPT other than the original plain text. The operation of XOR between the LPT and RPT gave a unique structure to the process and also helped to strengthen the algorithm. It is observed that the double XOR operation will increase execution time and also help to reduce the time taken in the final permutation. Therefore, it reduces the time that would be taken to do tedious permutation by the computer carrying out the encryption. DEA is a better choice where less memory is required but as memory becomes needed, the TDEA with DXOR becomes the best. The results show that the encryption time of DEA is less than TDEA and with the implementation of DXOR, the execution time becomes reduced with an increase in the strength of the algorithm. Figure 8 shows the triple data encryption algorithm used in this work.



```
for(int i = 0; i < 48; i++) {
right_expanded += right[expansion_table[i]-1];
};  // 3.3. The result is xored with a key
string xored = Xor(round_keys[i], right_expanded);
string res = "";
// 3.4. The result is divided into 8 equal parts and passed
// through 8 substitution boxes. After passing through a
// substituion box, each box is reduces from 6 to 4 bits.
for(int i=0;i<8; i++){
// Finding row and column indices to lookup the
// substituition box
string row1= xored.substr(i*6,1) + xored.substr(i*6 + 5,1);
int row = convertBinaryToDecimal(row1);
string col1 = xored.substr(i*6 + 1,1) + xored.substr(i*6 + 2,1) +
xored.substr(i*6 + 3,1) + xored.substr(i*6 + 4,1);;
int col = convertBinaryToDecimal(col1);
int val = substition_boxes[i][row][col];
res += convertDecimalToBinary(val);
}
```

Figure 8: Triple data encryption algorithm

Table 4: Data file size and memory size

DATA FILE SIZE(BYTES)	DEA	TDEA (3 KEYS)	TDEA (3 KEYS)+DXOR	TDEA (2 KEYS)	TDEA (2 KEYS)+DXOR
100	1200	1200	1200	1200	1200
200	1400	2000	1900	1600	1500
300	1200	1900	1900	1600	1500
400	1400	2800	2500	2000	1800
500	1200	2600	2500	2300	1800
600	1400	3000	2500	2300	1800
700	1400	3000	2500	2300	1800
800	1300	2300	2500	1900	1800
900	1300	2700	2200	2100	1400
1000	1200	2700	2200	2100	1400

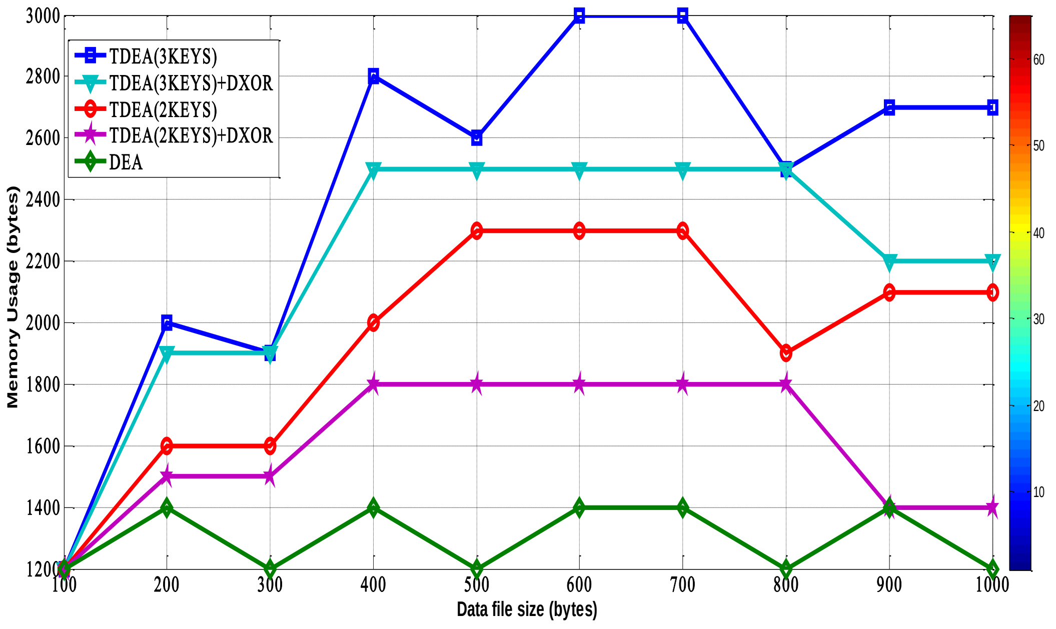


Figure 9: Plot of memory usage against data file size

Table 5: Average response time for data encryption algorithm

	Average response time (second)				
File size (MB)	**DEA**	**TDEA (3 KEYS)**	**TDEA (3 KEYS)+DXOR**	**TDEA (2 KEYS)**	**TDEA (2KEYS)+DXOR**
1	0.15	0.451	0.309	0.31	0.270
5	0.52	1.62	1.43	1.22	1.13
10	1.40	3.99	3.71	2.35	2.19
15	2.13	6.51	6.13	4.09	3.62
20	3.47	11.01	10.61	7.29	6.99

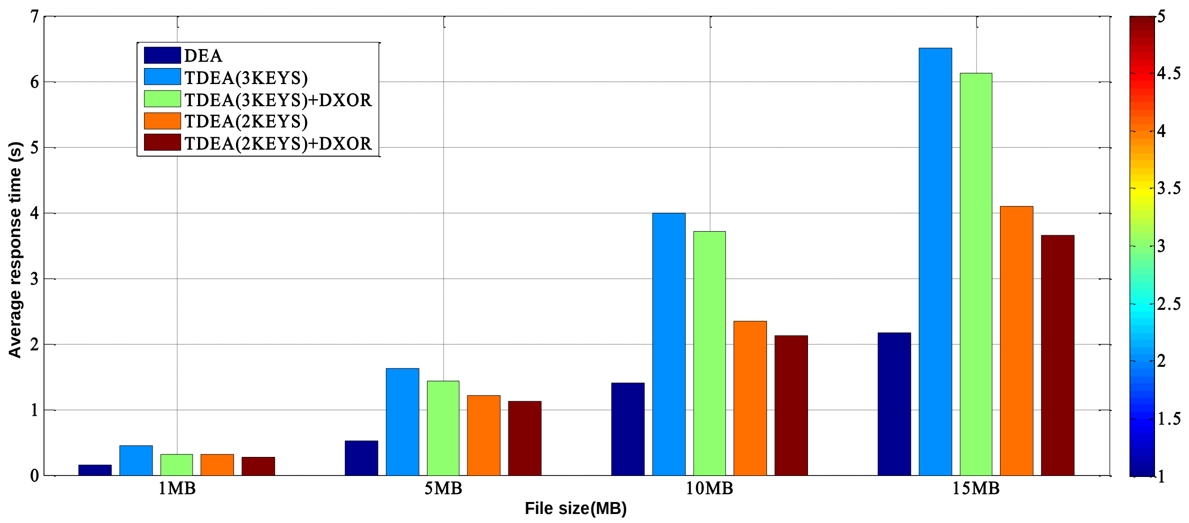


Figure 10: Bar chart showing average response time for data encryption algorithm at selected file size

The memory usage of each algorithm produces more performance at smaller memory sizes of 1200 to 3000 bytes for selected data sizes. Table 4 shows the memory usage against data file size for each of the algorithms during encryption. Figure 9 shows that more memory has been used during encryption by TDEA algorithm compared to DEA. The implementation of DXOR operation in the TDEA algorithm reduces memory usage during execution.

It was observed that the data encryption algorithm has a better response time as the file size increases. Also, when the security strength of the triple data encryption algorithm (TDEA) was compared with DXOR operation, the TDEA (2KEY) + DXOR was better than TDEA. It is also observed from the plot that at 1MB of file size, DEA had the lowest value of average response time of 0.2s while TDEA (3KEYS) had the highest value of 0.5s. Again, at 5MB, DEA had the lowest value of average response time of 0.5s while TDEA (3KEYS) had the highest value of 1.6s. Also, at 10MB, DEA had the lowest value of average response time of 1.4s while TDEA (3KEYS) had the highest value of 4.0s. Again, at 15MB, DEA had the lowest value of average response time of 2.1s while TDEA (3KEYS) had the highest value of 6.5s.

CONCLUSION

Triple Data encryption algorithm with double exclusive OR using two keys was the best option in applications where security is the major factor. DEA provides great performance if speed is required during encryption and decryption when using large data.

Based on the performance evaluation, the results of DEA, TDEA (3KEYS), TDEA (2KEYS), TDEA (3KEYS) + DXOR and TDEA (2KEYS) + DXOR were compared and it was observed that the optimized triple data encryption algorithm provide more security based on the availability of the resources.

However, based on the comparison between the symmetric block ciphers, the TDEA (2KEYS) + DXOR was the most suitable candidate for security and has the potential for further development due to a significant advantage in memory, encryption, and decryption time.

REFERENCES

Akshitha, V., Roshan, R.S., Nawaz, S., and Ravindra, J. (2020). An Efficient Optimization and Secured Triple Data Encryption Standard using Enhanced Key Scheduling Algorithm. Procedia Computer Science Direct of Elsevier. 171: 1054 – 1063.

Albrecht, M., Grassi, L., Rechberger, C., Roy, A., and Tiessen, T. (2016). Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. 22nd International conference on the theory and applications of cryptology and information security. 191-219.

Arunima, J., Gaurav, R., and Dheerenda, S. (2014). Security Testing of Web Applications: Issue and Challenges. International Journal of Computer applications. 88(3): 0975-8887.

Devanshu, B. (2018). Cyber Security Risks for Modern Web Applications: Case Study Paper for Developers and Security Testers. International journal of scientific and technology research. 7(5): 1-4.

Hamdan, O. A., Zaidan , B.B., Zaidan, A.A., Hamid, A. J., Shabbir, M., and Al-Nabhani, Y. (2010). New Comparative Study Between DES, 3DES and AES within Nine Factors. Journal of Computing. 2(3): 152- 157.

Marwa, E. S., Abdelmgeid, A.A., and Fatma, A.O. (2016). Data Security using Cryptography and Steganography Techniques, International

Journal of Advanced Computer Science and Applications. 7(6): 390 -397.

Mohamed, A., and Yousef, S. A. (2014). The Reality of Applying Security in Web Applications in Academic. International Journal of Advanced Computer Science and Applications. 5(10): 7-15.

Kaur, A. and Kumari, S. (2014). Secure Database Encryption in Web Applications. International Journal of Advanced Research in Computer and Communication engineering. 3: 7606-7608.

Kong, F. (2016). Research on Security Technology Based Web Application. Information Science and Management Engineering IV. 367-370.

Tiago, V. and Carlos, S. (2016). Web Applications Security and Vulnerability Analysis Financial Web Applications Security Audit - A Case Study. International Journal of Innovative Business Strategies (IJIBS). 2(2): 86-94.

Prapulla, S. B., Trisha, V., Jayanth C., Sindhu, B. D., and Vinhya, N. (2017). Mobile Agent Data Security using Triple Data Encryption Standard. International Journal of Applied Engineering Research, 12 (22): 11824 – 11829.

Smitha, N. M. (2018). Data Security in Cloud using Advanced Encryption Standard. International Journal of Engineering Research and Technology (IJERT). 7(1): 205 - 208.