International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 3 Issue 5, August 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Performance Analysis of Data Encryption Standard (DES)

Thida Soe, Soe Soe Mon, Khin Aye Thu

Faculty of computer systems and Technology, University of Computer Studies, Hinthada, Myanmar

How to cite this paper: Thida Soe | Soe Soe Mon | Khin Aye Thu "Performance Analysis of Data Encryption Standard (DES)" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-5, August 2019, pp.1439-1443, **https://doi.org/10.31142/ijtsrd26650**

IJTSRD26650

ABSTRACT

Information security is becoming much more important in data storage and transmission with the fast progression of digital data exchange in electronic way. Cryptography has come up as a solution which plays a vital role in information security system against malicious attacks. The cryptography is most important aspect of communications security and becoming an important building block for computer security. This security mechanism uses some algorithms to scramble data into unreadable text which can be only being decoded or decrypted by party those possesses the associated key. To protect sent messages that some of the most commonly used cryptography methods with private key based algorithm are LOKI (89, 91, 97), DES, triple DES, AES, Blowfish, etc. These algorithms also include several computational issues as well as the analysis of DES algorithm. The main features that specify and differentiate one algorithm from another are the ability to the speed of encryption and decryption of the input plain text. This paper analyzes the private key based algorithm DES and LOKI91 by computing index of coincidence (IC) and time efficiency.

KEYWORDS: *cryptography, DES, LOKI91, Index of Coincidence (IC), run time*

INTRODUCTION

Cryptography algorithms play an important role in providing security to networks. They can be categorized into symmetric (private) and asymmetric (public) keys encryption.

In asymmetric keys, two keys are used; private and public keys. Public key is used for encryption and private key is used for decryption. In symmetric keys encryption or secret key encryption, only one key is used to encrypt and decrypt data. Symmetric algorithms can be divided into two categories: block ciphers and stream ciphers. The block ciphers operate on data in groups or blocks. The most common secure algorithms are DESX, a variation of Data Encryption Standard (DES), Blowfish and Advanced Encryption Standard (AES), etc. The less common secure algorithms are IDEA, RC6, LOKI (89/91/97), etc.

For each algorithm there are two key aspects used: Algorithm type (define size of plain text should be encrypted per step) and algorithm mode (define cryptographic Algorithm mode). Algorithm mode is a combination of a series of the basic algorithm and some block cipher and some feedback from previous steps. LOKI uses one 64-bit keys, DESX uses one 128-bit key while AES uses various 128, 192 or 256 bit keys. Blowfish uses various 32-448 default 128 bits. Encryption algorithms consume significant amount of computing resources such as CPU time, battery power etc. We compare and analyzed algorithms DES and LOKI91.

A. Data Encryption Standard (DES)

DES is a block cipher. It encrypts the plaintext data in a block of 64 bits and produces 64 bit cipher text. The cipher key length is 56 bits and round key length is 48 bits. Initially the key is consisting of 64 bits with the parity bits 8, 16, 24, 32,40,48,56, 64 and that bits are used for parity checking and is ignored. DES is based on the fundamental attributes of cryptographic functions such as substitution (confusion) and transposition (diffusion), exclusive-OR operation, shifting and swapping. DES consists of 16 rounds and each round used only one round key that keys are generated from the round key generator. Round key generator crates the 48 bit keys for each round.

- **DES Algorithm**

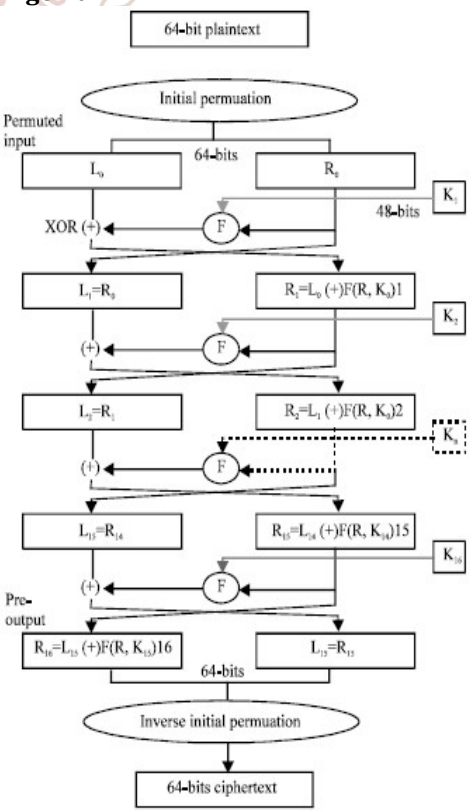

Fig.1 Schematic DES algorithm

The algorithms steps are as follow:

1. In the first step, the initial 64-bit plain text block is handed over to initial permutation (IP) function.
2. The initial permutation is performed on plain text.
3. The initial permutation produces two halves of permuted block: Left plain text (L_0) and Right plain text (R_0).
4. Each of L_0 and R_0 goes through 16 rounds of the encryption process, each with its own key.

The single round of DES is shown in Fig. 2. These steps are

A. From the 56-bit key, a different 48-bit sub-key is generated by using key transformation.
B. Using the expansion permutation, the R_0 is expended from 32 bits to 48 bits.
C. Now, the 48-bit key is XORed with 48-bit K_i and the resulting output is given in the next
D. step.
E. Using the S-box substitution produce the 32-bit from 48-bit input.
F. These 32 bits are permuted using straight P-box permutation.
G. The straight P-box output 32 bits are XORed with the $L_0$32 bits.
H. The result of the XORed 32 bits becomes the R_1 and old R_0 become the L_1. This process is called as swapping. Now the R_1 again given to the next round and performed the 15 more rounds.

5. After the completion of 16 rounds, the final permutation is performed.

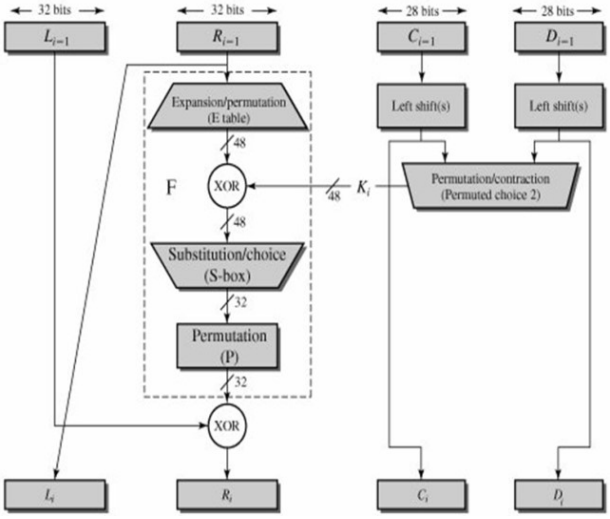

Fig.2 Single Round of DES

B. LOKI

In cryptography, LOKI89 and LOKI91 are symmetric-key block ciphers designed as possible replacements for the Data Encryption Standard (DES). The ciphers were developed based on a body of work analyzing DES, and are very similar to DES in structure. The LOKI algorithms were named for Loki.

➢ LOKI89

The cipher uses a 64-bit block and a 64-bit key. Like DES, it is a 16-round Feistel cipher and has a similar general structure, but differs in the choice of the particular S-boxes, the "straight P-permutation", and the "Expansion permutation". The S-Boxes use the non-linearity criteria developed by Josef Pieprzyk, making them as "complex" and "unpredictable" as possible. Their effectiveness was compared against the known design criteria for the DES S-boxes. The permutations were designed to "mix" the outputs of the S-boxes as quickly as possible, promoting the avalanche and completeness properties, essential for a good Feistel cipher. However unlike their equivalents in the DES, they are intended to be as clean and simple as possible, aiding the analysis of the design.

Following the publication of LOKI89, information on the new differential cryptanalysis became available. This resulted in the design being changed to become LOKI91.

➢ LOKI91

LOKI91 was designed in response to the attacks on LOKI89. The changes included removing the initial and final key whitening, a new S-box, and small alterations to the key schedule. More specifically, the S-boxes were changed to minimize the probability of seeing different inputs resulting in the same output (a hook which Differential cryptanalysis uses), thus improving LOKI91's immunity to this attack. The changes to the key schedule were designed to reduce the number of "equivalent" or "related" keys, which resulted in the exhaustive search space for the cipher being reduced.

Whilst the resulting cipher is clearly stronger and more secure than LOKI89, there are a number of potential attacks. Consequently these ciphers should be viewed as academic efforts to advance the field of block cipher design, rather than algorithms for use. The number of citations and published critiques suggests this aim has been achieved.

➢ LOKI91 Algorithm

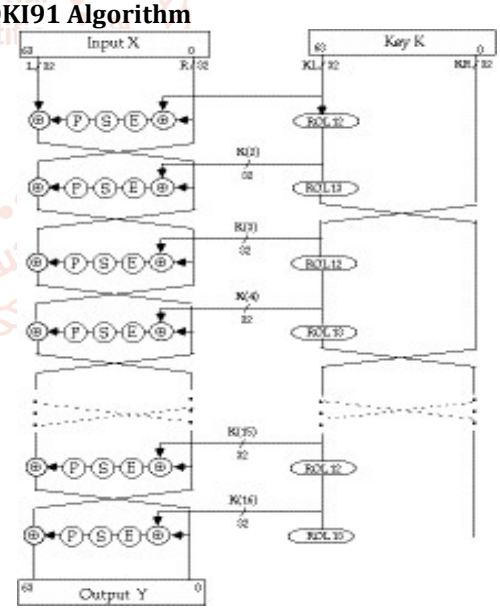

Fig.3 Schematic LOKI91 algorithm

The mechanics of LOKI91 are similar to DES. The data block then divided into a left half and right half and goes through 16 rounds, much like DES. In each round, the right half is first XORed with a piece of the key, then through an expansion permutation.

The 48 bit output is divided into four 12-bit blocks, and each block is sent through an S-box substitution. Then, the four 8-bits output are recombined to form a single 32-bit number and sent through the permutation. Finally, the right half is XORed with the left half to become the new right half, and the right half becomes the new left half. After 16 rounds, the block is again XORed with the key to produce the ciphertext.

The sub-keys are generated from the key in a straight forward manner. The 64-bit key is split into a left half and a right half. This left half is then rotated 12 or 13 bits to the left and then every two rounds the left and right halves are exchanged. As with DES, the same algorithm can be used for both encryption and decryption, with some modification in how the sub-keys are used.

C. Analysis of DES and LOKI91 Algorithms

➢ Implementation of DES Algorithm

The procedures of encryption program are as follow:

1. Read a key of any 8 characters.
2. Convert the key from ASCII to binary from, and then obtain 64-bits input.
3. Change the first 28 bits position by using the first of permuted choice (PC-1). Then we obtain C_i 28 bits and the last 28 bits for the second part, so we get D_i 28 bits.
4. We should check whether we could use a 1 circular left shift (1 rotate left) or two circular left shift (2 rotate left). After these operations, we concatenate the C_i and D_i then obtain CD_i 56 bits. Then change the CD_i position by using the permuted choice 2(PC-2), so we get K_i 48 bits.
5. After repeating this operation 16 times, we get 16 number of K_i.
6. Read the input file name and the output file name the operation.
7. If end of encounter, the process will stop.
8. Read the 8 characters from input files and convert the ASCII code to binary for, so we obtain the 64 bits input. Then change position by using initial permutation (IP).
9. The left 32 bits is called left permuted input L_0 and the right 32 bits is called right permuted R_0.
10. For J = 1 to 16.
11. L(J) = R(J-1)
12. This R(J-1) is expended from 32 bits to 48 bits by using the expansion table. The operate XOR calculation of this R(J-1) 48 bits and K(J) 48 bits and get new nr0 48 bits.
13. Exacts 6 bits from nr0. The first bit and the last bit represents the row and the middle 4 bits represents the column of selection function S_i. Then we get 4 bits from S_i and after this operation 8 times, we obtain 8 number of S_i. We concatenate S_1 to S_8 we obtain var32 bits.
14. Change positions in var32 by using the primitive function (P), and so we obtain 32 bits pvar32.
15. This pvar32 and L(J-1) are subjected to XOR calculation, thus obtain R(J).
16. Next J. After this operation, get 16 numbers of R(J) and L(J).
17. Concatenate the last position of R(16), then obtain preoutput RL64 bits.
18. Operate the initial permutation inverse (IP^{-1}) with RL. Then obtain the ciphertext 64 bits.
19. Write to a ciphertext 64 bits to output file.
20. Go to step 7.

➢ **Implementation of LOKI91 Algorithm**

The LOKI91 Algorithm is developed by window XP personal computer using visual C++ programming language.

1. Read a key of any 8 characters.
2. Convert the key from ASCII to binary from, and then obtain 64-bits input.
3. Change the first 32 bits position by using the first of permuted choice (PC-1). Then we obtain C_i 32 bits and the last 32 bits for the second part, so we get D_i 32 bits.
4. After the first round, the rotation of the left sub-key alternated between 12 and 13 bits to the left. Then every two round, the left and right halves are exchanged.
5. After repeating tis operation 16 times, we get 16 number of K_i.
6. Read the input file name and the output file name the operation.
7. If end of encounter, the process will stop.
8. Read the 8 characters from input files and convert the ASCII code to binary for, so we obtain the 64 bits input. Then change position by using initial permutation (IP).
9. The left 32 bits is called left permuted input L_0 and the right 32 bits is called right permuted R_0.
10. For J = 1 to 16.
11. L(J) = R(J-1)
12. This R(J-1) is expended from 32 bits to 48 bits by using the expansion table. The operate XOR calculation of this R(j-1) 48 bits and K(j) 48 bits and get new nr0 48 bits.
13. Divided into four 12 bit blocks. The two left-most bits and the two right-most bits represent the row and the middle 8 bits represent the column of selection function S_i. Then we get 8 bit from S_i and after this operation 8 times, we obtain 8 number of S_i.
14. Change positions in var 32 by using the primitive function (P), and so we obtain 32 bits pvar 32.
15. This pvar 32 and L(J-1) are subjected to XOR calculation, thus obtain R(J).
16. Next J. After this operation, get 16 numbers of R(J) and L(J).
17. Concatenate the last position of R(16), then obtain preoutput RL64 bits.
18. Operate the initial permutation inverse (IP^{-1}) with RL. Then obtain the ciphertext 64 bits.
19. Write to a ciphertext 64 bits to output file.
20. Go to step 7.

➢ **Index of Coincidence (IC)**

The index of coincidence IC, measures the variation in the frequencies of letters in the ciphertext. If the period of cipher is one (1), that is simple substitution has been used, there will be considerable variation in the letter frequencies of IC will be high. As the period increase, the variation is gradually eliminated (due to the diffusion) and IC is low.

Let the length of the text be $\boldsymbol{N}$ and let the size of the alphabet be $\boldsymbol{n}$. Consider the $\boldsymbol{i}$-th letter $\boldsymbol{a_i}$ in the alphabet. Suppose $\boldsymbol{a_i}$ appears in the given text $\boldsymbol{F_i}$ times. Since the number of $\boldsymbol{a_i}$'s in the text is $\boldsymbol{F_i}$, picking the first $\boldsymbol{a_i}$ has $\boldsymbol{F_i}$ different choices and picking the second $\boldsymbol{a_i}$ has only $\boldsymbol{F_i}$-1 different choices because one $\boldsymbol{a_i}$ has been selected. Since there are $\boldsymbol{N}(\boldsymbol{N}$-1) different ways of picking two characters from the text, the probability of having two $\boldsymbol{a_i}$s is

$$\frac{F_i(F_i-1)}{N(N-1)}$$

Since the alphabet has $\boldsymbol{n}$ different letters and the above applies to each of them, the probability of having two identical letters from the text is'

$$\sum_{i=1}^{n} \frac{F_i(F_i-1)}{N(N-1)} = \frac{1}{N(N-1)} \sum_{i=1}^{n} F_i(F_i-1)$$

Therefore, the index of coincidence is:

$$\mathrm{IC} = \frac{1}{N(N-1)} \sum_{i=1}^{n} F_i(F_i-1)$$

English has ***n*** = 26 letters.

D. Experimental and Simulation Analysis

We analyzed the two algorithms with English texts and these texts are selected randomly. And then input file sizes are used from small to large size. Different samples yield slightly different result. We analyzed the performance of these algorithms by computing Index of Coincidence (IC) and time efficiency.

TABLE1 ENCRYPTION AND DECRYPTION EXECUTION TIME

Input File Size(Bytes)	Encryption Execution Time(Seconds)		Decryption Execution Time(Seconds)	
	DES	LOKI91	DES	LOKI91
100	0.12	0.11	0.14	0.13
500	0.14	0.12	0.16	0.15
2000	0.15	0.13	0.17	0.14
5000	0.14	0.12	0.15	0.12
10000	0.16	0.14	0.18	0.15

TABLE2 INDEX OF COINCIDENCE (IC)

Input File Size(Bytes)	Index of Coincidence (IC) for Plaintext		Index of Coincidence (IC) for Ciphertex	
	DES	LOKI91	DES	LOKI91
100	0.959604	0.959604	0.067401	0.067522
500	0.955932	0.955932	0.081535	0.082189
2000	0.961995	0.961995	0.089604	0.090311
5000	0.976799	0.976799	0.102679	0.103218
10000	0.982129	0.982129	0.097445	0.100812

TABLE 1 represents the five different sizes of files and corresponding encryption and decryption execution time taken by DES and LOKI91 algorithms in seconds. By analyzing the TABLE 1, we conclude that the encryption and decryption time taken by LOKI91 is slightly small as compare to DES. These comparisons are shown in Fig.4.

TABLE 2 represents the five different sizes of files and corresponding Index of Coincidence (IC) by DES and LOKI91 algorithms. By analyzing the table 2, we conclude that the IC for ciphertext by LOKI91 is large as compare to DES. These comparisons are shown in Fig.5.

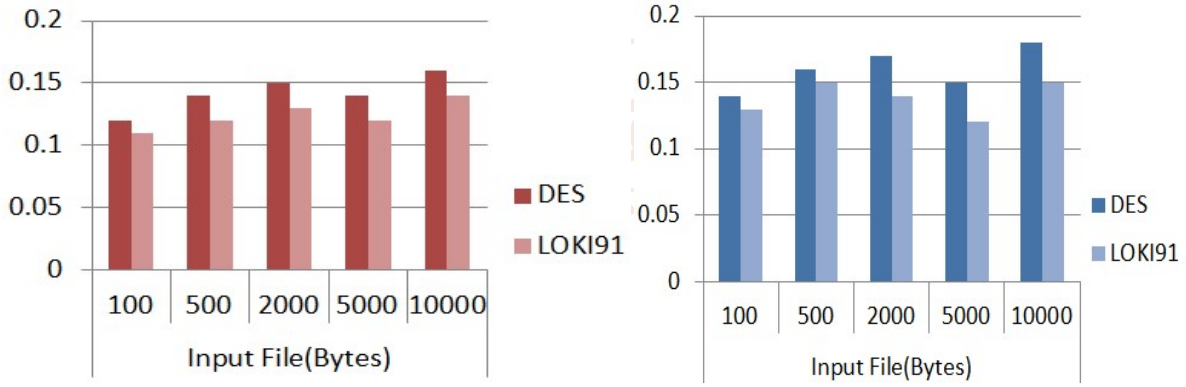

Fig. 4 Execution Time for (a) Encryption (b) Decryption

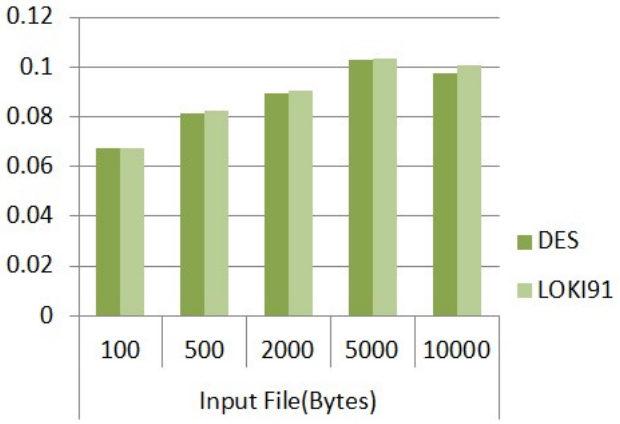

Fig. 5 Index of Coincidence (IC) for Ciphertext

E. CONCLUSIONS

This paper presents the performance evaluation of cryptographic algorithms for various types of files. we have studied that the encryption and decryption execution time consumed by DES algorithm is not quite different for some sort of message compared to LOKI91 algorithm. Index of Coincidence (IC) for the ciphertext by DES is smaller than the LOKI91. Thus, the performance of DES is very good as compared to LOKI91.

References

[1] Rishabh Arora Sandeep Sharma, PhD, "Performance Analysis of Cryptography Algorithms", International Journal of Computer Applications (0975 – 8887) Volume 48– No.21, June 2012.

[2] Sombir Singh[1] , Sunil K Maakar [2] and Dr. Sudesh Kumar3, "A Performance Analysis of DES and RSA Cryptography", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 3, May – June 2013.

[3] Abdullah Al Hasib, Abul Ahsan Md. Mahmudul Haque," A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography", Third International Conference on Convergence and Hybrid Information Technology,2008.

[4] William Stallings, "Cryptography and Network Security Principles and Practices." Prentice Hall, November 16, 2005.

[5] A. Nadeem, "A performance comparison of data encryption algorithms", IEEE information and communication technologies, pp.84-89, 2006.Bn

[6] Gaurav Shrivastava, "Analysis Improved Cryptosystem Using DES with RSA." VSRDIJCSIT, Vol. 1 (7), 465-470, 2011.

[7] Charels Connell, "An Analysis of New DES: A Modified Version of DES", Locust Street Burlington, USA, Boston MA 02215 USA.