REPUBLIC OF TURKEY
FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND
APPLIED SCIENCES

LIGHTWEIGHT IMPLEMENTATION OF DES FOR RESOURCE CONSTRAINED DEVICE

MUKHLIS IBRAHEM MUHAMAD SHARIF

Master Thesis

Department: Software Engineering

Supervisor : Assist. Prof. Dr. Fatih OZKAYNAK

June-2018

REPUBLIC OF TURKEY

FIRAT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

LIGHTWEIGHT IMPLEMENTATION OF DES FOR RESOURCE CONSTRAINED DEVICE

MASTER THESIS

MUKHLIS IBRAHEM MUHAMAD SHARIF

(152137107)

Submission Date to the Institute: 18/6/2018

Thesis Presentation Date: 11/6/2018

Thesis Supervisor : Dr. Fatih ÖZKAYNAK (F.U.)

Other member of the jury : Prof. Dr. Asaf VAROL (F.U.)

Assoc. Prof. Dr. Fatih TALU (I. U.)

June-2018

DECLARATION

I hereby declare that the thesis entitled "Lightweight Implementation of DES for Resource Constrained Device" is my own research and has been prepared by myself, except for the passages and single words that are quoted. It is being submitted for a Master's Degree in Software Engineering at Fırat University.

Sincerely

Mukhlis Ibrahem Muhammad Sharif

ELAZIĞ-2018

DEDICATION

I am dedicating this thesis work to all my beloved ones, especially my wife who has always been a constant source of support and encouragement during my thesis work. I am also dedicating it to every single person who is interested in knowledge and new inventions.

ACKNOWLEDGEMENTS

First of all, I am very grateful to Almighty Allah, who enabled me to complete this work with full devotion.

I wish to express my sincere gratitude to all my instructors at (First University) for their assistance, support and help through the college process to access this stage. My sincere gratitude and appreciation to my friendly supervisor Dr. Faith ÖZKAYNAK Assistant Professor in Software Engineering Department (First University) for his invaluable support and guidance throughout this thesis work.

I would also like to show my particular gratitude to Firat University Scientific Research project Unit for supporting my thesis work whose project number is TEKF. 18. 02.

Of course, I shouldn't forget to express my warmest thanks to my parents, wife and all family members. And all my friends who always supported my works, without their encouragement I wouldn't have reached my dream.

Sincerely

Mukhlis Ibrahem Muhammad Sharif

ELAZIĞ-2018

TABLE OF CONTENTS

Page. N

DECLARATION ..... **I**

DEDICATION ..... **II**

ACKNOWLEDGEMENTS ..... **III**

TABLE OF CONTENTS ..... **IV**

ABSTRACT ..... **VII**

ÖZET ..... **VIII**

LIST OF TABLES ..... **IX**

LIST OF FIGURES ..... **X**

ABREVIATIONS ..... **XI**

1. INTRODUCTION **1**

1.1. Literature Review 3

1.2. Aim of the Thesis 8

1.3. Statement of the Problem 8

1.4. Proposed Approach 9

1.5. Method of Study 10

2. DES ALGORITHM **11**

2.1. Important Corner Stones in the Historical Development of the DES Algorithm 12

2.2. Overview of DES Algorithm 14

2.3. Feistel Block Cipher 15

2.4. General Algorithm of DES 15

2.5. DES Algorithm in More Details 16

2.6. Encryption Operation on DES Algorithm 20

2.6.1. Initial and Invers Permutations (input 64-bits) 20

2.6.2. Round Function 23

2.6.3. Expansion P-Box 23

2.6.4. Exclusive Disjunction/Exclusive OR (XOR) 24
2.6.5. Substitution Boxes (S-Boxes) 25
2.6.6. Permutation Box (P-Box) 28
2.7. Key Operations on DES Algorithm 29
2.7.1. Generating the Round Key 30
2.7.2. Key Permutation-1 30
2.7.3. Binary Left Rotation 31
2.7.4. Contraction Permutation 31
2.8. Decryption Operation on DES Algorithm 32

3. FAST SOFTWARE IMPLEMENTATION OF DES FOR RESOURCE CONSTRAINED DEVICE 35
3.1. Lightweight Cryptography 35
3.2. Design Approaches Lightweight Ciphers 35
3.3. Proposed Fast Software Implementation of DES 37
3.4. Analysis and Experiment Results of Proposed Method 41
3.5. Discussion 41

4. GENERATION ALTERNATIVE SBOX STRUCTURES BASED ON CHAOTIC SYSTEM 42
4.1. Importance of S-Box 42
4.2. Why do we need to Generate Alternative Structures S-Box? 42
4.3. Chaos-Based S-Box Design Studies 43
4.4. Details of Proposed Method 44
4.5. Analysis and Test Results 48
4.6. Discussion 51

5. CONCLUSION AND FUTURE WORK 52
5.1. Conclusion 52
5.2. Future Work 52

REFERENCES 53

CURRICULUM VITAE 59

ABSTRACT

Lightweight Implementation of DES Algorithm for Resource Constrained Devices

Many services are being moved to digital environments with industry 4.0. However, the information in the digital environment poses a great risk. Therefore, ensuring the security of information on the users of online applications is a necessity. This perspective requires secure signal processing. But it is difficult to guarantee secure signal processing for resource-constrained devices. Throughout history, many cryptographic algorithms such as DES, 3DES, AES have been suggested to transferring sensitive information. But there are several problems in these algorithms such as the security, speed, excessive consumption of energy, the internal structure of the design are not clear to the public. Lightweight cryptology has emerged to solve some these problems.

This thesis presents new design architecture for improving the software implementation of the DES. New proposed acceleration technique has been used to speed up the DES algorithm in software. The results show that the developed code is faster than the original DES. Another contributor to the thesis is a chaos-based substitution box. The performance metrics of the proposed substitution boxes are close to the performance values of the original DES substitution boxes. Therefore, the new cryptographic building block is an alternative to original design. In fact, the design of the original DES substitution boxes is unclear and this is a problem. The clearness of the design of the new proposed substitution boxes aims to solve this problem.

Keywords: Information Security, Lightweight Cryptography, DES, Fast Software Implementation of DES Algorithm, Substitution box, Chaos.

ÖZET

Kaynak Kısıtlı Cihazlar için DES Algoritmasının Hafifsıklet Gerçekleştirimi

Endüstri 4.0 ile birlikte birçok servis sayısal ortamlara taşınmıştır. Fakat sayısal ortamlardaki veriler büyük riskler içermektedir. Bu yüzden çevrimiçi uygulama kullanıcı verilerinin güvenliğinin sağlanması bir zorunluluktur. Bu perspektiften güvenli veri işleme gereklidir. Fakat kaynak kısıtlı cihazlar için güvenli veri işlemeyi garanti edebilmek zordur. Tarih boyunca hassas bilgileri iletebilmek için DES, 3DES ve AES gibi birçok kriptolojik algoritma önerilmiştir. Fakat bu algoritmalarda güvenlik, hız, aşırı enerji tüketimi, tasarımın içyapısının açıkça bilinmemesi gibi çeşitli problemler bulunmaktadır. Hafifsıklet kriptoloji bu problemleri çözmek için ortaya çıkmıştır.

Bu tezde DES algoritmasının yazılım gerçekleştirimini iyileştirmek için yeni bir tasarım mimarisi sunulmuştur. Yeni önerilen hızlandırma tekniği, yazılımda DES algoritmasını hızlandırmak için kullanılmıştır. Sonuçlar geliştirilen kodun orijinal DES algoritmasına göre daha hızlı olduğunu göstermiştir. Çalışmanın bir diğer katkısı ise kaos tabanlı yerine koyma kutularıdır. Önerilen yeni yerine koyma kutularının performans ölçütleri orijinal DES yerine koyma kutularının performans değerlerine yakındır. Dolayısıyla, yeni kriptolojik yapıtaşı orijinal tasarıma bir alternatiftir. Orijinal yerine koyma kutularının tasarım mantığı açık değildir. Bu bir problem oluşturmaktadır. Önerilen yeni yer değiştirme kutusunun tasarım mantığının açık olması bu problemi çözmeyi hedeflemektedir.

Anahtar Kelimeler: Bilgi güvenliği, hafifsıklet kriptoloji, DES, DES algoritmasının hızlı yazılım gerçekleştirimi, yerine koyma tabloları, kaos.

LIST OF TABLES

Page No.

Table 2. 1. Initial Permutation……...….. 21

Table 2. 2. Final Permutations……..... 22

Table 2. 3. Expansion Permutation……...……..... 24

Table 2. 4. Xor Operations…….……..... 24

Table 2. 5. S-boxes Operation……..... 28

Table 2. 6. P-Box …….....…… 28

Table 2. 7. Key P_1…….....…… 30

Table 2. 8. Bit Shifts…….....……...…… 31

Table 2. 9. Contraction Permutations……..…….....…… 31

Table 3. 1. Result comparison between DES and it's opponent…….....……...…… 41

Table 4. 1. Performance Value for classical DES S-box Structure….....……...…...… 49

Table 4. 2. Performance Value of proposed S-box Structure….....……...…… 50

LIST OF FIGURES

Page No.

Figure 2.1. Principle of diffusion of a block cipher.....12
Figure 2. 2. Encryption and Decryption Process on DES Algorithm.....14
Figure 2.3. General structure of DES 16
Figure 2.4. Encryption operations 18
Figure 2.5. Initial and Final Permutation.....20
Figure 2.6. Round Functions 23
Figure 2.7. S-Box array 25
Figure 2.8. S-Box Rules.....25
Figure 2.9. Key Operations.....29
Figure 2.10. Decryption Operations 34
Figure 3.1. Space and security comparisons.....36
Figure 3. 2. Performance comparisons for software implementation.....37
Figure 3. 3. Performance comparisons for power consumption.....37
Figure 3. 4. Overview of DES block cipher 38
Figure 3. 5. Proposed Design Architecture.....40
Figure 4. 1. Codes used to generate alternative DES's S-box structures 45
Figure 4. 2. Flowchart of roadmap of constructing an alternative S-boxes.....46
Figure 4. 3. Generated Proposed S-box structures 48

ABREVIATIONS

DES	Data Encryption Standard
AES	Advanced Encryption Standard
IP	Initial Permutation
IP^{-1}	Final Initial permutation
IBM	International Business Machine
NSA	National Security Agency
NBS	National Bureau of Standard
NIST	National institutes of standard and technology
FIPS	Federal Information Processing Standard
CBC	Cipher Block Chaining
OFB	Output Feedback
LWC	Lightweight Cryptography
RFID	Radio Frequency Identification
ASIC	Application-Specific Integrated Circuit
FPGA	Field Programmable Gate Array
IC	Integrated Circuit
IoT	Internet of Things
GA	Genetic algorithm
SAC	Strict Avalanche Criterion
BIC	Bit Independence Criterion

1. INTRODUCTION

Cryptography is an ancient art of obscuring sensitive information that is transmitted between sender and receiver. This process had started in Egyptian days about 4000 years ago. However, it had a crucial role in the outcomes of First and Second World Wars. Encryption in the past was synonymous with its word but today it's based on computer science and mathematical theory [1]. The rapid improvements in the technology (communications systems and computers) in 1960's led to a claim a special science in order to protect digital information and provide security, and in 1970's IBM started working on this request through Feistel structure with making some changes on it to become the most widespread standard which is Data Encryption Standard (DES). There are two types of cryptography, Symmetric cryptography and Asymmetric cryptography, in the first type for encryption and decryption process only one key used, but in the second type each process has its own key [2].

The Data Encryption Standard (DES) is the powerful standard for ciphering data and it’s known as a symmetric algorithm. However, now DES is replaced by a new standard known as the Advanced Encryption Standard (AES) because of some problems in DES. DES encrypts data of 64-bit (8-byte) at a time that means it’s a block cipher of 64-bit (8-byte). It’s unlike stream cipher that encrypts only one bit at a time (or encrypts a small groups of bits such as a byte in some time) [3].

The National Bureau of Standards (NBS), currently known as National Institute of Standard and Technology (NIST) sent a request to Federal Register to create an encryption algorithm with some world standard such as (more secure, easy for understanding….) in 1973. In the beginning of 1974 International Business Machines (IBM) suggested commercializing LUCIFER with some of important changes that were introduced on 23 Nov 1976 to become the Data Encryption Standard (DES). The most significant of this change was the size of key, DES uses key size of 56-bit instate of 128-bit key size used in LUSIFER. In spite of that DES uses a 56 bit key as input the other 8 bits are used for parity checking and have no effect on DES’s security .The key of 56-bit is simple to be broken by brute force attack because it has a very small size. IBM was not by any means the only one involved in these changes as they looked for specialized guidance from the National Security Agency (NSA). The change

adaptation of LUCIFER was advanced as a proposal for the new national encryption standard asked for by the National Bureau of Standard (NBS) [4].

The S-boxes that are used were designed secretly and no reasons were given for their particular design. The early discovery was that the S-boxes which seemed to be secure against Differential Cryptanalysis attack which was only publicly discovered by Biham and Shamir in 1990 after 13 years. Actually the designers of DES said that the cause behind not making a specified design for the S-boxes was that the number of attacks weren't known by the public at that time and they didn't want any leakage. Anyhow, in spite of all this controversial issue, NIST reaffirmed DES for the government in 1994 for other five years in order to use it in other areas [5].

Finally, in 1977 the NBS released the modified IBM cipher with all specifications as the Data Encryption Standard (FIPS PUP 46) to the public. Even though the cipher is described down to the bit level in the standard, the motivation for parts of the DES design called (design criteria), specially the choice of the substitution boxes were never officially released [1]. In the early 1980's the pair was increased in personal computer and all DES specifications were available to the public, the inner structure of the cipher was easier to be analyzed. During this period, the research community of civil cryptography also grew and DES underwent major security. However, until 1990 no dangerous weak points were seen. Honestly, DES was standardized for 10 years until 1987. Because the DES was widely used and had security weaknesses, so the NIST reaffirmed the federal use of the cipher until 1999, when it was finally replaced by the Advanced Encryption Standard (AES) [6].

1.1. Literature Review

When we want to send our information (file/documents) through some devices such as (cell phones, Fax, laptops, etc.) by the internet as a soft copy using different communication instruments (like infrared, wireless, Bluetooth, etc.) safely we need to use some encryption algorithm. The most common algorithm throughout history to make our sensitive information unreadable and misused by anyone and only readable to the right person is Data Encryption Algorithm DEA [7]. Because DEA is an old algorithm today and the technology has a rapid development so it's not secure anymore. Therefore, we need to find an efficient solution to enhance encryption process of DES algorithm (like security and speed) issues to transfer our important information faster and more secure at the same time. So, we have made some improvements to DES algorithm such as designing new architecture of DES to accelerate it in software and designing new s-box based on the chaotic system to make secure communications for transferring our sensitive information [8].

In 1970 DES was designed and implemented by IBM and in 1977 it was accepted as a draft by National Institute of Standard and Technology NIST (which was known as the National Bureau of Standard NBS) for USA government applications. DES supports two inputs to the encryption operation: one is the plaintext of 64-bit and the other is the key of 56-bit in length with using 8-bit as a parity bit and it would need to 2^56 attempts to find the correct key [9]. Many researchers and cryptographers are working in many fields of DES encryption/ decryption operations to make it better.

Singh *et al.* [10] tried to enhance DES algorithm by changing the size of the input block data by extended it three times to become 192-bit in order to decrease the number of the cycle of cipher/decipher operations, and increase the sub-key size twice to become 128-bit. The EDES that they designed is more resistible to many cryptanalytic attacks but takes more time and is slower than the original one.

The authors in [11] mixing some robust modern symmetric block cipher such as Feistel, DES, and AES designed a powerful secure encryption algorithm based on some encryption principles such as CPU Usage, Encryption and Decryption Time, Avalanche Effect, and throughput. They attend a hybrid structure of Feistel/DES, Feistel/AES, and DES/AES to design

robust encryption techniques that are resistant to efficient attack like brute force attack, meet in the middle attack, differential cryptanalysis, etc. of course this hybrid structure is very powerful in security but it doesn't solve the main problems of original DES like unclear the inner structure (s-box structure) and DES's speed in lightweight platforms.

In the work of [12] used the concepts of the Data Encryption Standard with some changings in size of entered block data and in key size to fast DES in the software without regardless some important point in the DES (like security, clearly, etc.). In their new design algorithm using block data of 32-bit as input with the key of size 32-bit, on the other hand the entire operations (Permutation, Expansion, substitution) is the same as original DES operations. Instead of using 32-bit of 64-bit of entered block data in the original DES they use all input block data of 32-bit with the key of 32-bit. So, fastDES uses multiplication concept instead of subtraction that is used in DES (i.e. the operational data is less than the entered data in original DES on the contrary of fastDES the operational data is greater than the entered data). Of course by using block data of 32-bit it's faster and only used in that area that where speed is important thing. It is faster than DES but it is also so easy for attackers to hack this fastDES because both entered block data and key size are short.

The authors in [13] also present new design architecture of DES algorithm for mobile and hoc network (MANET) to solve the energy consumption problem due to the encryption algorithm. This proposed algorithm strengthens DES algorithm so as to consume less energy by reducing the number of rounds from 16-rounds to 8-rounds and increasing the size of key twice to become 112-bit (use two keys of 56-bit in size). It's better than the classical DES algorithm in security, consumption of energy and resistible to many attacks according to their experimental results and using some encryption principles (like Less Energy Consumption through Limited computation, Brute force attack, Avalanche effect, Key Management & Complexity, etc.) but it also can't reach that level to consider it secure to be compared with a modern encryption algorithm.

A study in [14] using the concept of fusion techniques they are accurately read three modern algorithms (DES, Blowfish, and Genetic algorithm) and take the Magic Thoughts from the techniques of designing these three algorithms to design an optimal proposed fused DES-Blow algorithm. By using the concept of DES algorithm and taking the future of a

generation key in blowfish and obtaining the good Genetic algorithm (GA) idea to generate a strong key. It has some characteristics like securely, understandable, publicity, etc. Fused DES-Blow is a symmetric block cipher with 64 input bit and use two different right key and left a key in agenda. By applying the idea of generating the key in blowfish algorithm generate the right key of 48-bit and generating the left key of 48-bit by using genetic algorithm depending on the hamming distance to protect the key from any weakness. Of course, by using two different keys for encryption/decryption process this increases the complexity to the attacker to get the key from the cipher text and should try 2^2n if the key length was n. This algorithm improves the security problem but at the same time compounded the problem of speed and not mentioned the designing of entire structure problem that is not clear to the public.

The work of [15] also depended on the notion of the fusion techniques to generate the strict key for Data Encryption Algorithm. In their work, three proposals have been offered depending on a dynamic pool that contains many random bits that come from the key that is used previously and other resources for increasing the seed of key generated randomly, and these pools are updated when the key is getting from pool and add the key by the user. The first approach used proposed an artificial neural network according to some step and used hamming distance as a comparing criteria for weak key to generate a random keys, in the second proposal genetic algorithm is used according to the Hamming distance function to generate a random key, and finally used the conception of Blowfish for generating sub-keys by modifying their method of generating the keys. Mohammed *et al*. [16] enhanced the Data encryption algorithm sub-key. After explained some weak point of DES's key (like if all 64-bit are equal to 0's and 1's, If all right and left half of key become 1's and' 0's in the case the key for all round becomes the same, and some other weakness of classical DES's key like 0x0101010101010101, 0xE0E0E0E0F1F1F1F1, etc.). Their enhance is generating sub-keys by using genetic algorithm depending on the primary key to generate different set of pseudorandom sub key when program executed. In the same context, Arrag *et al*. [17] attended an algorithm that extends the creation and expansion operations of the key encryption process of the Advanced Encryption Standard algorithm.

The authors in [18] providing some enhance in DES algorithm and called modified

DES by changing something in the heart part of DES algorithm which is the Function (the change is on the right half of 32-bit after expanded it to 48-bit by E and divided this 48-bit into two block of 24-bit) the objective of this division is to more resistant against the differential cryptanalysts. Also to increase the complexity against the exhaustive attack and the time memory trade-off they are two keys of (128-bit) in size are used. Through their experimental results, they have improved security of DES but also there is some unsolved problem remains in their modified DES.

The authors in [19] they have improved the DES algorithm structurally to design new strong algorithm based on the DES characteristics called the Improved-DES. They extended the input block of data from 64 bits to 93 and divided into 3 sub-blocks of 32 bits and for each sub-blocks used different functions, and then increase the S1-58 of the S-boxes to 51-516 and S-boxes are enlarged to sI-sl6, so as to satisfy some cryptography criteria such as SAC, and correlation coefficient. Also, extend the key of DES from 64 bits to 112 bits.

Above where some researches about how to enhance DES and many of them have displayed how to accelerate DES software and ignore IBM trapdoor, which is the eternal structure of DES (S-box's structure was not clear to the public until nowadays). In the past few decades, the chaotic system become the hot topic for enhancing the communication system's futures, the following works provide some solutions for s-box structure problem.

Zakaria *et al*. [20] they are worked on delete AES weak point (which is s-box) by providing new design architecture to enhance the security of AES algorithm; the most common part of AES to make it more secure is substitution boxes. In their research, they suggested a new design of s-box based on an affine transformation function and adding a new function based on crossover and mutation process, these two way can meet the requirements confusion and diffusion characteristics in cryptography. For testing the security of their new design approach of s-box they used the algebraic attack.

The authors in [21] worked to provide a secure design for a lightweight system such as Radio Frequency Identification (RFID) by modifying one of the most common modern block ciphers which is (DES) with consideration to cost and hardware implementation. Instead of using eight S-boxes in classical DES they used one S-box in their modified DES (DES Lightweight) eight times to reduce the gate complexity for hardware implementation based on

DES's s-box criteria, and applied key-whitening so as to resist against some attackers like brute force attacker.

Seberry *et al*. [22] provided a powerful method for designing robust m*n s-box that withstand some cryptographic criteria. The method is based on group Hadamard matrixes, that gives the immunity to linear cryptanalysis, and satisfy some of newer encryption criteria (like SAC, balanced, etc.). Similarly, Detombe *et al*. [23] provided a novel way to construct robust 5*5 s-box based on the Boolean near-bent function used five odd variables. It differs from bent function in balance, it's not 0 1 balanced.

In the last fewer years, the easy changeable dynamic systems with impressive to primary conditions which are (chaotic maps) become a hot topic for designing robust random S-box [24]. Many researchers go around some of these maps such as (exponential map, Tent map, Lorenz system, logistic-sine map, Baker Map, HorseShoe Map, fractional-order chaotic Chen system, Cat Map, etc.) to generate good S-box that satisfies the encryption properties such as (nonlinearity, bijective property, output bits independence criterion, SAC, and equiprobable input/output XOR distribution, BIC, etc.). The researches, Özkaynak *et al*. [25] proposed a new algorithm for designing an 8*8 robust s-box that satisfied some encryption properties. These new s-boxes are based on continuous-time chaotic Lorenz system instead of chaotic map and shifting operations added to the algorithm. This algorithm has a powerful resistant to encryption criteria such as (SAC, nonlinearity, bijective property, strict avalanche criterion, and BIC, etc.) more than any other algorithms because of using shifting operations on columns and rows.

Also many other researchers like (Özkaynak *et al*. [26]; Cavusoglu *et al*. [27]; Liu *et al*. [28]; Belazi *et al*. [29]; Lambić D *et al*. [30]; Özkaynak in [31]) and many other researchers used the chaotic maps.

1.2. Aim of the Thesis

1- Secure applications for the Internet of Things (IoT) are constantly increasing and many of them require some lightweight cryptographic algorithms. Most lightweight cryptographic algorithms were not designed to be efficient in software platforms. As a result, the throughput in software of these algorithms is low on recent IoT devices [32, 33]. The first goal of this thesis is to present a new design architecture for improving the software implementation of the DES. The DES is a family of block ciphers. It is efficient in hardware but its design was not oriented for software platforms. Aim of our algorithm is that to find a block cipher architecture design for lightweight applications.

2- The DES algorithm is an important encryption protocol that has influenced many cryptology studies. It is still used today in many practical applications such as an electronic passport. But another passive point is the creating designs of the internal DES's structure and the designing of S-boxes. Despite published DES, the design criteria were protected unclearly to the public until 1994. Thus, the internal structure of the DES is questionable to the user and it was free of unobserved points of forgery that would be used to decrypt the messages by NSA without depending on the key [1, 33]. The second goal of this thesis is to proposed alternatives to DES s-box structures to solve this problem. The analysis results of the propose structures are equivalent to DES s-box structures.

1.3. Statement of the Problem

DES is a family of block ciphers. It is efficient in hardware but its design was not oriented for software platforms [34]. Aim of our algorithm is that to find a block cipher architecture design for lightweight applications.

There have been two approaches for designing of lightweight ciphers [35, 36]. These approaches are:

- Optimized low-cost implementations for standardized and trusted algorithms such as DES, AES.
- Design new ciphers with the goal of having low hardware implementation costs.

Even though both approaches are valid, these approaches have some problems that we will explain in chapter 3.

Also the mathematical background of DES substitution boxes is not cleared until nowadays it's questionable to the cryptographers. We solve this problem by designing S-boxes that have mathematical background.

1.4. Proposed Approach

The reason for this selection of DES is that it is suitable lightweight hardware implementation, if we compare the one-round implementation of AES and DES; the latter consumes about 6% of the logic resources of AES [37]. In this thesis, two new approaches are proposed for enhancing DES in (security and speed) sides.

Of course, it's not easy to create a new design architecture that consists of all these properties (secure, take a small space of memory and very fast) at the same time because these operations coincide with one another. A different operation such as mod, XOR, s-box, and permutation are required for greater security, but these processes both reduce speed and increase area cost. An optimum design approach of DES proposed for a lightweight application would be to have a well-investigated encryption by implementing atomic operation through Table Look-up (time-memory trade-off). Three of the available techniques can be used to speed up the software implementation of the DES algorithm [38].

- Merging permutation.
- Realize bit permutation by table look-up.
- Realize S-box substitution by table look-up.

Firstly: The first technique is used in this work to design an optimum approach of DES. **Secondly:** the entire structure of DES S-box was not clear mathematically to the public [33]. In this work, an alternative DES's S-box is proposed based on chaotic maps.

1.5. Method of Study

The DES is "a school for generating cryptographers"; it is an important encryption protocol that has influenced many cryptology studies. It is still used today in many practical applications such as an electronic passport. But it has an important problem; the design parameters of substitution boxes of this algorithm are unknown. For this reason, we have proposed alternatives to DES's S-box structures to solve this problem. The analysis results of the proposed structures are equivalent to DES s-box structures.

In this research, we have enhanced the DES algorithm into two sides (speed side and security side).

- **Side 1:** We provide new design architecture to accelerate DES algorithm for lightweight platforms. In the classical DES algorithm there is a stack data structure (first in last out) with E and P in the DES function, in our modified approach using merging bit permutation (releasing the permutation after substitution and putting it before expansion) to solve the stack problem and accelerate DES in software. They are the first explorers of this problem in DES algorithm. According to their experimental results, they did a great discovery in the DES algorithm.
- **Side 2:** The most worrisome side in cryptologic terms is s-box. Substitution box tables of the DES algorithm have been examined. The rationale design of DES s-box structures is unknown, and then we improved new design architecture to generate s-box structures that can be used as an alternative to the DES algorithm.

2. DES ALGORITHM

There are two types of encryption data, encryption data bit by bit (stream cipher) and encryption data by grouping (block cipher). DES is a block cipher that means it operates on the block of clear (plain) text. It is one of the most particular symmetric-key block ciphers. It supports two inputs to the encryption operation which are the plaintext of 64-bit and the key of 56-bit in length with using 8-bit as a parity bit and it would need to 2^56 attempts to find the correct key. In DES encryption the sender and recipient need only one key, i.e., DES is a symmetric key encryption, and for encryption and decryption, the plain text uses the same key of 56-bit in length [39]. The DES algorithm takes input plain text of 64-bit of fixed length and process using the key, and the plain text is transformed through a difficult chain of operations to produce the cipher-text (of the same length). Actually, the DES uses the key of 64-bit in length, but they are considered only 56-bit and the rest of 8-bit is used as a parity bits (for calculating checksum) [3]. The Data Encryption Standard (DES) is considered to be lacking length for many applications; even at present DES is not secure, then some standards analysis has theoretically proved that the DES algorithm is weak. Since the best example for illustrating symmetric algorithm is DES algorithm, its design principles inspired many of the newer ciphers [40].

In old-fashioned algorithms, the attacker was able to get the key and the plaintext within the cipher-text because the relationship between the cipher-text and the plaintext was somehow clear. For instance, there are some alphabetical letters in English that are repeated more than other letters, in this case, the attacker could find out the key plaintext because there was a clear relationship between cipher-text and plaintext. There were also some other ways; the attacker can use this information to break a cryptographic algorithm. In 1945 Claude Shannon published in his paper under the name of Mathematical Theory of Cryptography two important principles for a secure cipher operation, which made an algorithm to be more powerful in encryptions called Diffusion and Confusion [41].

Claude Shannon invented Confusion process in order to make the statistical relationship between plain text and encryption key complicated as much as it could so as to avoid the attempts of figuring out the key [41].

This illustrates that each change in the key affects all other characters in the cipher-text block. This relation will reduce the effect of attacker's statistics of the cipher-text. Even if the attacker knows the statistics of the cipher-text, it is still hard to figure out the key.

Using a difficult substitution algorithm can represent confusion. Nowadays, the most commonly applied confusion is a substitution, which is found in both DES and AES. We can represent confusion as ancient substitution like Caesar's cipher and new substitution like Feistel network which substitute each bit using lookup table. Look at the (S-Box) in table below [42].

A B C…….Z

D E F……..C in this classical substitution the key is cyclic shifted three times.

A B C…….Z

E Y O…….W in this modern substitution there are factorial of 26 possible keys.

For example, look at the figure 2.1 the change of 1-bit in the plain text must perfectly effect on average in the half of the output bits.

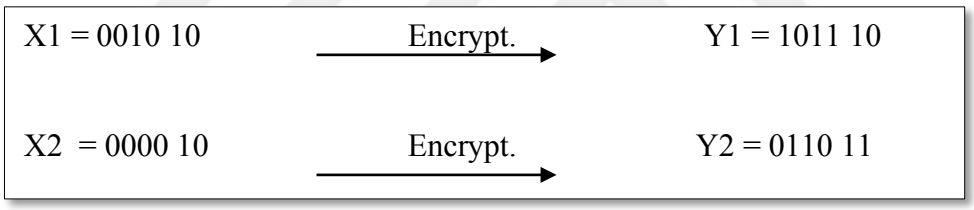

Figure 2.1. Principle of diffusion of a block cipher

2.1. Important Corner Stones in the Historical Development of the DES Algorithm

The world wars and national security issues led to security in communications, this was in the past. But now security issues have reached businesses and private sectors. E-Commerce is in need to a secure Internet communications that's why many corporations have firewalls in order to protect their information from other competitive corporations. Personal privacy, in private sector is raising restlessness. Products could be sold by many means of communications such as e-mails and telephone calls. We can provide secure communications by encryption. Encryption makes data not understandable to the third party. This data can be not understandable by inverse of this operation, which is called decryption by the recipient.

For providing a secure communication, many present electronic communications use encryption operation [43]. The most widely used standard for encryption data is Data Encryption Standard (DES). Rapid improvements in technology have adversely affected the security of the DES algorithm and found out some weak points in DES.

One of the obstacles of DES is that we can't consider it secure because the key size is so small which is 56-bit that needs to try out about 936 possible keys that would be out of the question because computers couldn't have enough speed to do that and it makes DES vulnerable to brute-force attackers [44]. Some traditional non-DES encryption algorithms have been made to remove this problem such as 3DES, AES and many other encryption algorithms. These algorithms are robust as a tank, but the great challenge to this algorithm is the consumption of energy in our smart devices such as mobiles and laptops. Because these traditional encryption algorithms require considerable power to transfer and process sensitive data and they are going beyond the challenge of security and speed, but there is a problem of Consumption of exceeded energy in our electronic devices batteries. Here are some worrying statistics about consumption of energy in our devices. There were 9.5 billion devices in year 2005 each one of them consumes 160 Mw the total consumption energy = 2.8€ billion mw in the year. According to the Cisco estimates connected devices in 2020 will become 50 billion and Intel says that it will become 200 billion smart devices by 2020. Ok, what’s the solution? The solution becomes the lightweight cryptography [45].

Also DES has software implementation (speed) issues, technically the DES standard is hardware standard. That means, the encryption structure is as per the standard just when it is executed in a physical electronic circuit. DES software implementation refers to implementation of DES on desktop CPU's or embedded microprocessor like cell phone and smart card. DES hardware implementation refers to implementation of DES on integrated circuit (IC's) such as application-specific integrated circuit (ASIC's) or field programmable gate arrays (FPGA's) Integrated circuit (IC's). DES is very fast in hardware. Some operations of DES like Expansion Box, Permutations, Initial permutation, and Inverse Initial Permutation are easy to implement in hardware because it doesn’t need logical operators, it only needs wiring connection, likewise, it’s easy to implement substitution boxes in hardware because they require Boolean logic. We need nearly 100 gates for each S-Box. The most difficult

operation of DES algorithm to be implementing in hardware is DES rounds, because each round can be done with less than 300 gates. Also DES was not designed for software and relatively runs fast in software. DES consists of many parts like table and operations that makes it hard to be implemented in software [46].

The other issue of DES is the creating designs of the internal DES's structure and the designing of S-boxes. Despite published DES, the design criterion was protected unclearly to the public in 1994. Thus, the internal structure of the DES is questionable to the user was it free of unobserved points of forgery that would be used to decrypt the messages by NSA without depended on the key? As well, the design that S-boxes are formed is not clear to the public and it's questionable to the user that why S-boxes have not mathematical background? Is this weak point used by IBM as a backdoor? All these questions and uncertainties revolve around our minds and the minds of the users. In this study, we have been proposed new robust S-Boxes to remove all these uncertainties and designing secure system by using a logistic equation as a mathematical background based on chaotic Lorenz system [1, 33].

2.2. Overview of DES Algorithm

The Data Encryption Standard (DES) is a block cipher that ciphers a block of data of 64-bits in length with a key of 56-bits in size as well as shown in Figure 2.2. This process is the same for the decryption operation [34].

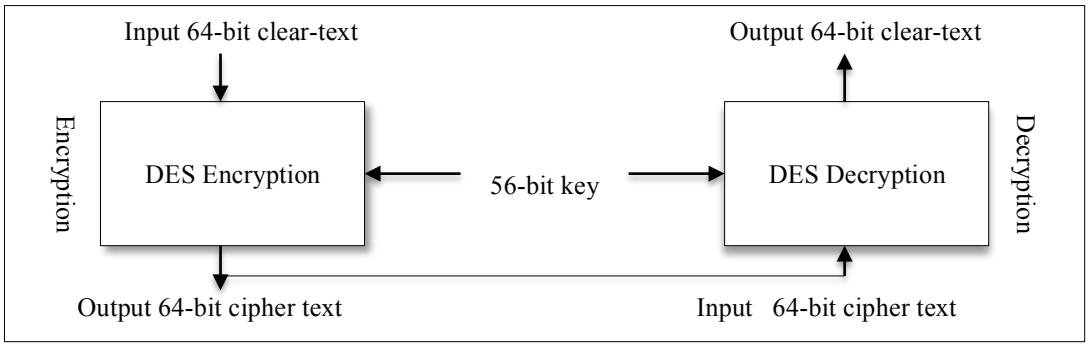

Figure 2.2. Encryption and Decryption Process on DES Algorithm

In DES encryption and decryption operation, the same sub-key that means the DES is a symmetric-key is used. The National Institute of Standards and Technology (NIST) published

DES. DES depends on Feistel Cipher in its operations. There are sixteen rounds in its operations; each round is detached that means the different sub-key is used for each different round. General algorithm of DES is graphical in the following illustration [47, 48]:

- Initial and final permutation.
- Round function.
- Key schedule.
- Any additional processing.

2.3. Feistel Block Cipher

Feistel structure uses a sequence of repeated encryptions on block of data and it is mostly created for block ciphers that encrypt large amounts of data. A Feistel structure works by dividing a block of data into two equal parts and stratify encryption in multiple rounds. Each round executes permutation and combinations derived from the main function or key. The number of rounds is different from each cipher that applied a Feistel structure. DES is just one example of a Feistel Cipher. Further, a cryptographic operation depending on Feistel structure uses the same algorithm for both encryption and decryption [48].

2.4. General Algorithm of DES

Nowadays, the most widely used algorithm is DES algorithm because it is secure to some extent. It is so difficult to break the DES because it uses a key length of 56-bit block cipher. There are seventy quadrillion probable keys of 56 bits. These algorithms should be created in a way that they could be applied in a computer system or network in order to supply cryptographic protection to binary encrypted data. The method of implementation will depend on the application and environment. Figure 2.3 shows a series of events that happen through an encryption process in DES algorithm and shows the main idea about what is happening in this event. There are two permutations that held in the encryption process, called initial and final permutations, and sixteen Feistel rounds. The block data that are entered into the initial permutation are 64-bit. Each round uses a different 48-bit round sub-key; DES has an effective

key length of 56 bits. The encryption algorithm doesn't use the rest 8-bit of the 64-bits of the key (parity bits) [49]. General structure of DES is shown in the following figure (Fig. 2.3).

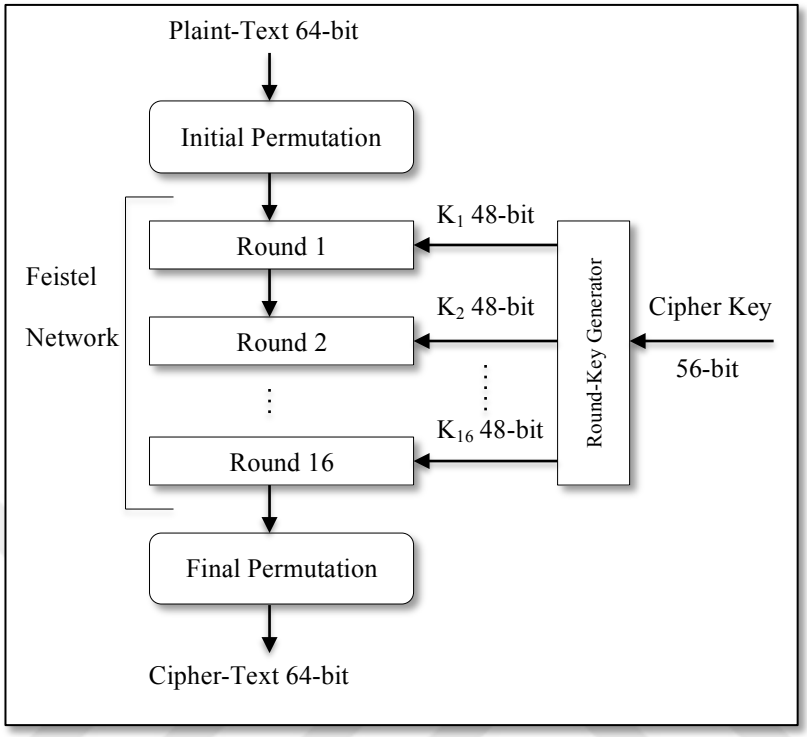

Figure 2.3. General structure of DES

2.5. DES Algorithm in More Details

The algorithm of DES is created to encode and decode a block of data comprising of 64-bits with key of a 64-bit. There are two main operations in DES algorithm encryption and key operations. The process of decryption in DES algorithm should be accomplished by utilizing a same key for encryption process, yet with the timetable of addressing the key bits changed so that the decoding procedure is the invert of the encryption procedure. Blocks of data (plain-text) that are to be coded are submitted in to an initial permutation (IP), to make a key more complex and finally it is submitted to final permutation which is the inverse of the initial permutation IP^{-1}, also this block is submitted in another operation we will give more details about it in next pages. The process of generating keys is also submitted to some operations (key permutation1, binary left rotation, and key permutation 2) to create a sub-key

of 48-bit in length. All the primitive operations used in DES can be split into two stages: the initial step is called encryption/decryption operations and the second step is called key operations [50]. Each of these operations is detailed in Figure 2.4.

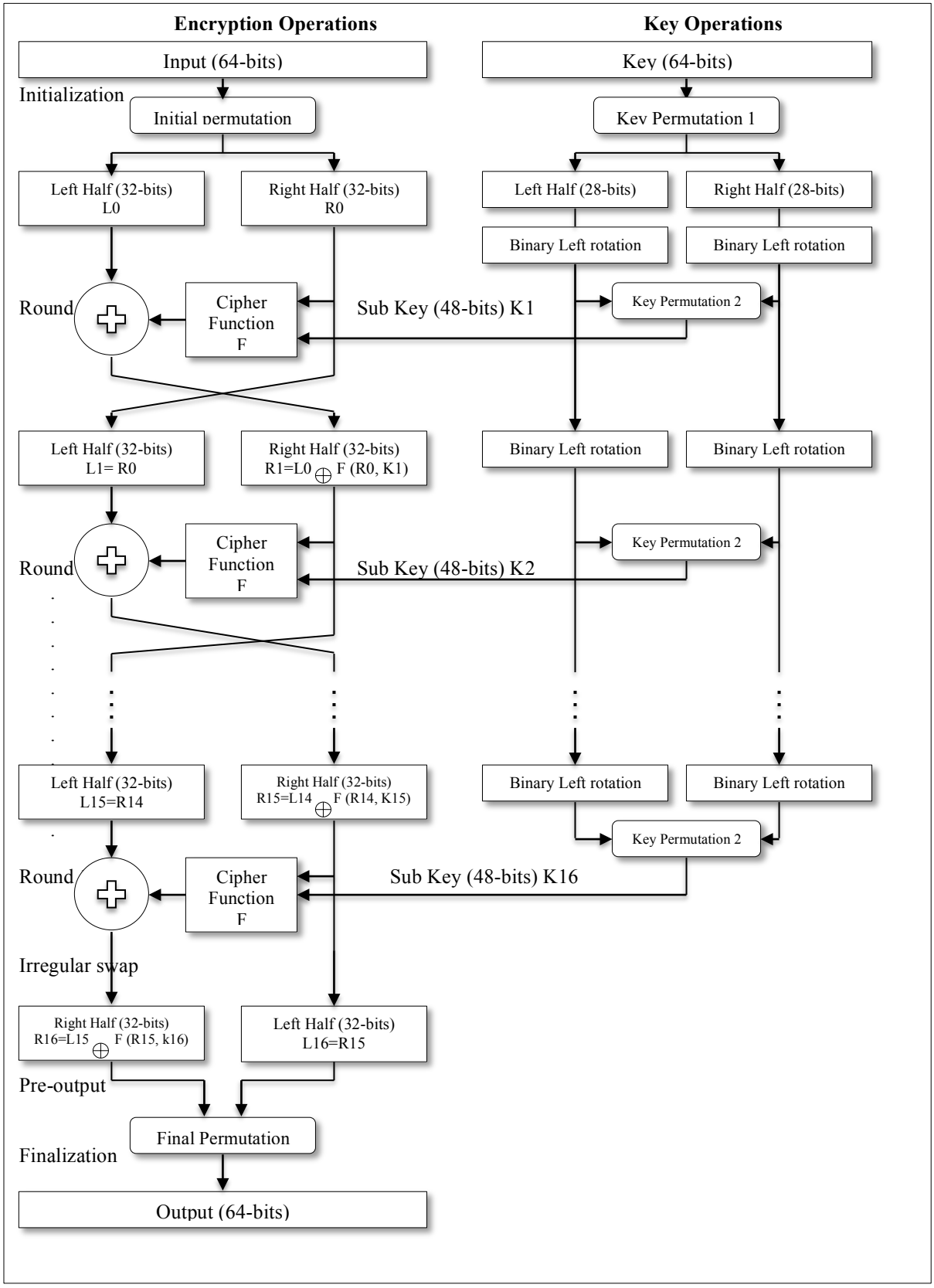

Figure 2.4. Encryption operations

If we look at the figure 2.4 it is divided into two parts, the Input text of 64-bits on the left side that is encryption operation, and key of 64-bits on the right side is the key operation. Here we will explain the left side of the Figure 2.4 (input text) that is encryption operation, and then we will explain the right side of the figure that is key operation, the plain text process submits in to three stages [51].

1. The plaintext of length 64-bit goes during an initial permutation (IP) that revamps the bits to output the permuted input. This is trailed by a stage consisting of 16 rounds of the same function, which includes both permutation and substitution operations.
2. The 64-bit that are produced by the last sixteenth round are mixture of the entered clear-text and the key, after the final sixteenth round both the left and right are swapped to create pre-output.
3. Lastly, the pre-output is gone during a final permutation (IP^{-1}).

The mission of initial permutation is to create non-arranged 64-bit cipher text. Actually, if we separate initial and inverse permutation from DES algorithm, in this case DES will become the same as structure of Feistel network. The right side of Figure 2.4 illustrates the process of generating different sub keys of 64-bit of input key, these bits are gone through key permutation 1 to become 56-bit and the 8-bit that are lost from 64-bit of key are used as parity bit. Then the output 56-bit of key permutation 1 are divided into two half of 28-bit. Then, different sub-keys (Ki) are created by the combination of binary left rotation and permutation key 2 for each of the different 16 rounds. The key permutation 2 processes are similar for each round [52]. We can give more details on DES algorithm in some mathematical operations as below:

- $IP\ (X)\ \Rightarrow\ L_0R_0$
- $L_i\ =\ R_{i-1}$
- $R_i = L_{\ i-1} \oplus\ F(R_{i-1}\ , K_i)$
- $Y = IP^{-1}(R_{16}\ , K_{16})$

- Note that, as usual:

- $R_{16} = L_{\ 15} \oplus\ F\ (R_{15}, K_{16})$
- $L_{16} =\ R_{15}$

- The final output of sixteenth round is swapped.

2.6. Encryption Operation on DES Algorithm

There are many parts that are used in encryptions of DES process that make the DES slower in the implementation on hardware and software [53].

2.6.1. Initial and Invers Permutations (input 64-bits)

The first 64-bit block of real data are gone through initial permutations after they go through other parts of DES encryption operations, after these operations the 64-bit output from the final round-16 are gone to the inverse permutation, and we can call the final permutation as straight Permutation boxes the first one is inverse to another. They have no cryptography significance in DES [54]. The initial and final permutations are shown in the Figure 2.5.

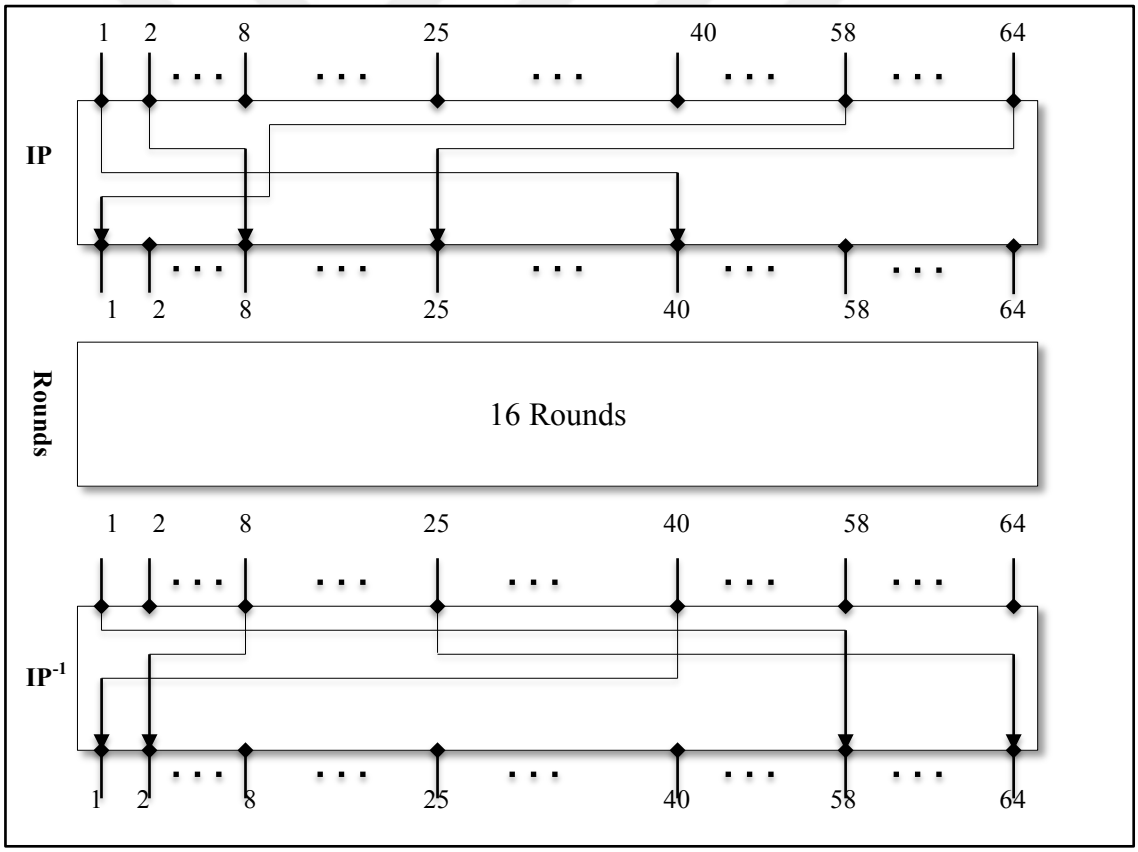

Figure 2.5. Initial and Final Permutation

DES's security is affected by the initial and final permutations they only order data. Both permutations are used to ease loading of clear text and cipher text data into a DES chip in byte-sized parts. It is difficult to use bit-wise in software as well as nonsense in hardware, that's why initial and final permutations are not dealt with in many software implementations.

At first the block of input 64-bit are submitted to the permutation below, which is called initial permutation IP.

IP

Table 2.1. Initial Permutation

Bits	Goes to position							
1-8	58	50	42	34	26	18	10	2
9-16	60	52	44	36	28	20	12	4
17-24	62	54	46	38	30	22	14	6
25-32	64	56	48	40	32	24	16	8
33-40	57	49	41	33	25	17	9	1
41-48	59	51	43	35	27	19	11	3
49-56	61	53	45	37	29	21	13	5
57-64	63	55	47	39	31	23	15	7

If we look at the Table 2.1 it has 8 rows and 8 columns, the first row consists of bit-1 to bit-8 and second row consists of bit-9 to bit-16, and so on. This table shows that the bit 58 of the entered data that goes through initial permutation becomes the first bit of the output, as well as bit 50 becomes the second bit of the output, and so on. Until we reach the last row that consists of bit-57 to bit-64 from the block of 64 bits of the clear text. The bit-7 of the entered data is considered as last bit (bit-64) after the initial permutation process. The output of the initial permutation is divided into right side of 32 bit and left side of 32 bits and goes through the 16 rounds until it reaches the final permutation [1].

The output of the last round (round-16) will be entered to the final permutation. Final permutation is opposite to initial permutation as explained in the Table 2.2.

IP^{-1}

Table 2.2. Final Permutations

Bits	Goes to position							
1-8	40	8	48	16	56	24	64	32
9-16	39	7	47	15	55	23	63	31
17-24	38	6	46	14	54	22	62	30
25-32	37	5	45	13	53	21	61	29
33-40	36	4	44	12	52	20	60	28
41-48	35	3	43	11	51	19	59	27
49-56	34	2	42	10	50	18	58	26
57-64	33	1	41	9	49	17	57	25

If we look at the Table 2.2 it has 8 rows and 8 columns, the first row consists of bit-1 to bit-8 and second row consists of bit-9 to bit-16, and so on. This table shows that the bit 40 of the entered data that goes through final permutation becomes the first bit of the output, as well as bit 8 becomes the second bit of the output, and so on. Until we reach the last row that consists of bit-57 to bit-64 from the block of 64 bits of the output of the last round. The bit-25 of the entered data is considered as last bit (bit-64) after the final permutation process. The output of the final permutation becomes an encrypted text of 64-bits [38].

2.6.2. Round Function

Figure 2.6 shows a single round of processing in DES Algorithm.

$L_i = R_{i-1}$, $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

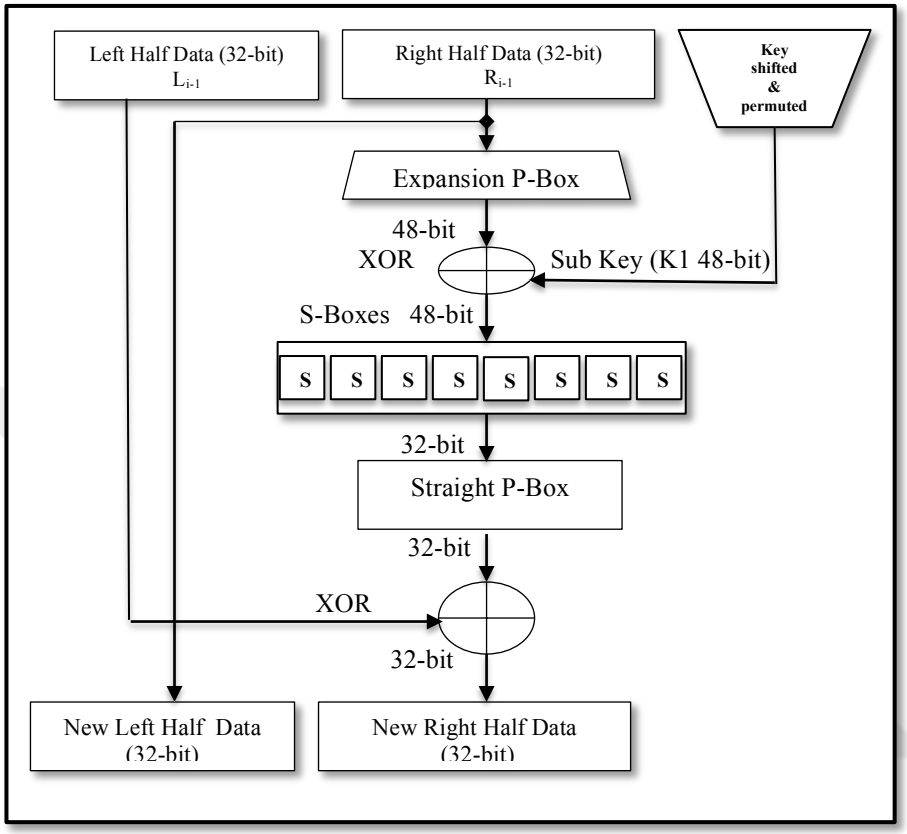

Figure 2.6. Round Functions

2.6.3. Expansion P-Box

If we look at the Figure 2.6 the sub key that is XORed with the right half data is 48-bit and the right half data is 32-bit. To XOR the right data half with the sub-key both of them should be same in size, and then we need an E-Box (Expansion Functions) that makes the input in size 32-bits of block of data and produces the output of size 48-bit of block of data [47].

DES uses the Table 2.3 to illustrate the Expansion process. Look at the data inputted into the E-Box is 32-bit but the output is 48-bit, because this table contains 8-columns and 4-rows that consist of only 1 to 32 cells but some of the input bits go further to more than one output [47]. For example, the value of input bit 1 goes to the value of output bits 2 and 48 as shown in Table 2.3.

Table 2.3. Expansion Permutation

Bit	1	2	3	4	5	6	7	8
Moves to Position	(2, 48)	3	4	(5, 7)	(6, 8)	9	10	(11, 13)
Bit	9	10	11	12	13	14	15	16
Moves to Position	(12, 14)	15	16	(17, 19)	(18, 20)	21	22	(23, 25)
Bit	17	18	19	20	21	22	23	24
Moves to Position	(24, 26)	27	28	(29, 31)	(30, 32)	33	34	(35, 37)
Bit	25	26	27	28	29	30	31	32
Moves to Position	(36, 38)	39	40	(41, 43)	(42, 44)	45	46	(47, 1)

2.6.4. Exclusive Disjunction/Exclusive OR (XOR)

After the expansion operation, the XOR process is used in DES on the right half data and the sub-key. The sub-key is used only in this process. The outputs of 48-bits that are produced by the E-Box are XORed with the sub-key of 48-bits, where indicates bit-by-bit addition modulo 2. This is indicated to as key mixing [52].

Table 2.4. Xor Operations

Inputs		Outputs
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.4 explains Exclusive OR operation, it is like the ADD operation that takes two inputs (0 or1) and it produces only one output (0 or 1). If both inputs are 0's or 1's the output will become 0, otherwise the result will become 1.

2.6.5. Substitution Boxes (S-Boxes)

After sub key is XORed with the expansion permutation, the obtained outputs of 48-bit are gone through substitution-Boxes (S-Boxes). As we can see in Figure 2.7 there are eight S-Boxes used in DES algorithm, the number of bits that are entered into each S-box are 6-bit and transform these 6-bit into the 4-bit as output, where the number of the input is not equal to the output. Note, for each S-Box the fixed tables should be used. The substitution represents confusion operation in DES algorithm, and it is the only non-linear element used in DES algorithm. (In this case we need the memory of size 256 bytes for eight DES S-boxes). The 48-bit inputs are divided into 6-bit sub-block [1, 47].

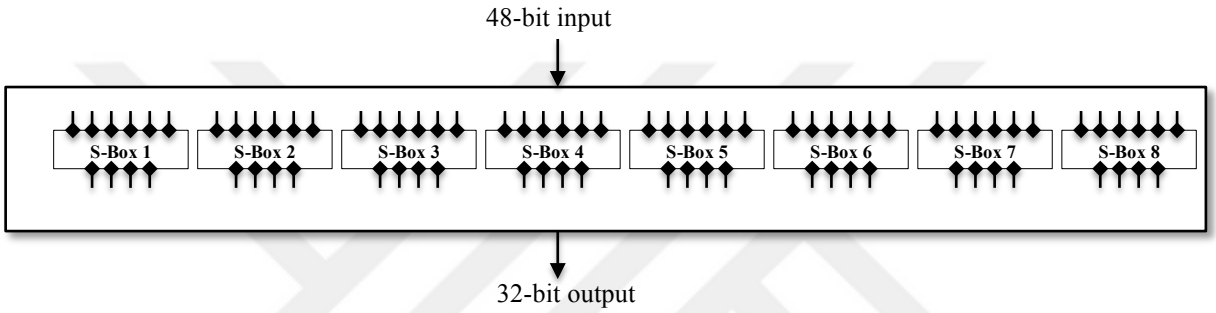

Figure 2.7. S-Box array

The Figure 2.8 illustrates the process of inverting 48-bit into 32-bit within substitution process, if we note that the first six bits of 48-bit are converted into 4-bit which is done in this way the first and last bit of the first six-bit represent the rows which consist of four rows. As well as the mid four bits represent the columns, which consist of sixteen columns [2].

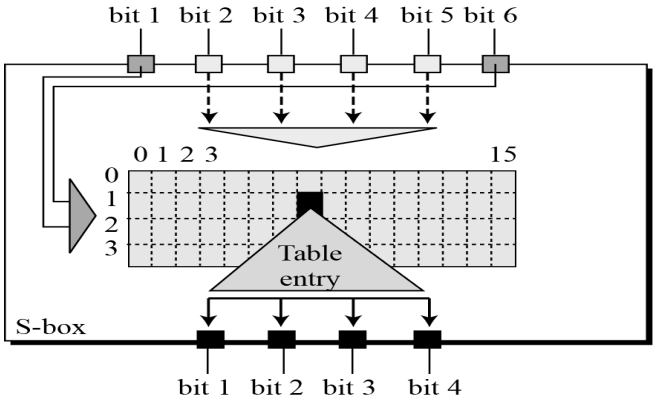

Figure 2.8. S-Box Rules

The output of second process of length 48-bit is divided into eight boxes, for each box the entered data is 6-bit, and each box produces only 4-bit, when these boxes reduced the data result will become 32-bit text. The substitution in each box follows a table based on 4-rows and 16- columns. The first 6-bit part is substituted by S-box 1, and the next 6-bit part is substituted by S-box 2, and so on. For example, if we enter the 011001 to s1, the row represents the first and the last bit of the entered value, 0 is the first bit and 1 is the last bit then the 01 is the row (row 1). And the middle 4-bit represent the column of S-box table, then the 1100 is the middle 4 bit that represent the column (column 12), and the value in row 1, column 12 is 9, so the output is 1001.

Table 2.5. S-Boxes Operations

S1

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01(1)	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10(2)	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11(3)	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
01(1)	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
10(2)	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
11(3)	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
01(1)	13	7	0	9	3	4	6	10	2	18	5	14	12	11	15	1
10(2)	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
11(3)	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
01(1)	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10(2)	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
11(3)	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	4	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
01(1)	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
10(2)	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11(3)	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
01(1)	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
10(2)	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
11(3)	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
01(1)	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
10(2)	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
11(3)	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)	1100 (12)	1101 (13)	1110 (14)	1111 (15)
00(0)	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
01(1)	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
10(2)	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
11(3)	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

2.6.6. Permutation Box (P-Box)

The final step in the round function is labeled Permutation Box (P-Box), the output of the S-Boxes is 32-bit and it is subjected to the final straight permutation in the round function with the same general rule in Table initial permutation (Table 2.1). For example, the sixteenth bit of the P-Box input becomes the first bit of the output as shown in the Table 2.6 [1, 47].

Table 2.6. P-Box

Bit	Goes to Position							
1-8	16	7	20	31	29	12	28	17
9-16	1	15	23	26	5	18	31	10
17-24	2	8	24	14	32	27	3	9
25-32	19	13	30	6	22	11	4	25

2.7. Key Operations on DES Algorithm

If we look at the Figure 2.9 it illustrates that there are sixteen of different 48-bit sub-keys that are generated by the key operations of 56-bit.

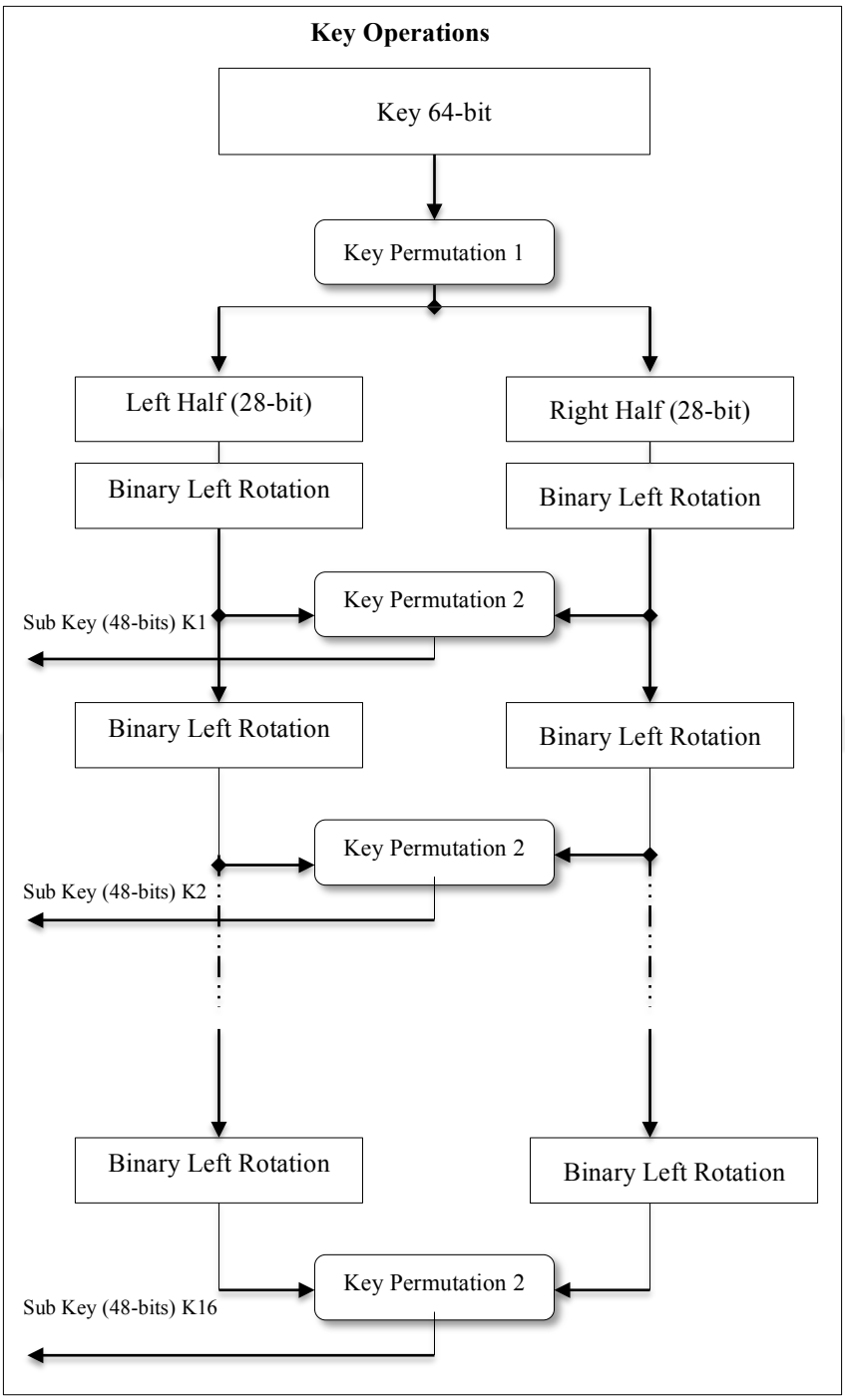

Figure 2.9. Key Operations

2.7.1. Generating the Round Key

After we finished explaining the encryption process at the left side of the Figure 2.4 now we will explain the right side of that figure that's called Key operation. There are many parts that are used in key operation in DES process.

2.7.2. Key Permutation-1

The size of key that is used in DES is 64-bit, which contains key of 56-bit in size and parity bit of size 8-bit. The last bit of each input 8-bit (1-byte) is parity bit. Then Key Permutation-1 is used to remove the parity bit from the input key of 64-bit in size. So this permutation gives 56-bits as output. Parity bits (namely, bits 8, 16, 24, 32, 40, 48, 56, 64), they don't appear in Table 2.7 and they are deleted from the input 64-bit key and the remaining bits are rearranged according to Table 2.7 [55].

Table 2.7. Key P_1

Bit	Goes to Position						
1-7	57	49	41	33	25	17	9
8-14	1	58	50	42	34	26	18
15-21	10	2	59	51	43	35	27
22-28	19	11	3	60	52	44	36
29-35	63	55	47	39	31	23	15
36-42	7	62	54	46	38	30	22
43-49	14	6	61	53	45	37	29
50-56	21	13	5	28	20	12	4

The table 2.7 illustrates that the 57th bit of the input becomes the 1st bit of the output, the 49th bit of the input becomes the 2nd bit of the output, and so on, until finally we reach the 56th bit of the output.

2.7.3. Binary Left Rotation

After permutation key-1, the key is divided into two parts (Right half and Left Half). Each part is 28-bit and left shift operation (left rotation) is 1 or 2 bits. In key rounds 1, 2, 9 and 16 shifting is only 1-bit, and in the other rounds, shifting is 2-bits. After that both of them are combined into the form of a 56-bit. The numbers of shifts are illustrated in Table 2.8.

Table 2. 8. Bit Shifts

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

2.7.4. Contraction Permutation

Contraction permutation process is applied in the same way as key permutation-1 in different permutation, it comes after the left shifting operation, and this operation combines both left and right sides of the output produced by the binary left rotation, which create a block of 56-bits. They are rearranged and contracted to form a round key of 48-bits (Table 2.9) [50].

Table 2.9. Contraction Permutations

Bit	Goes to Position							
1-8	14	17	11	24	1	5	3	28
9-16	15	6	21	10	23	19	12	4
17-24	26	8	16	7	27	20	13	2
25-32	41	52	31	37	47	55	30	40
33-40	51	45	33	48	44	49	39	56
41-48	34	53	46	42	50	36	29	32

If we look at the Table 2.9 we are missing 4-bit from the left half (bits 9, 18, 22, 25) and 4-bit from the right half (bits 35, 38, 43, 54), and it illustrates that the 14th bit of the input goes to the 1st bit of the output, the 17th bit of the input goes to the 2nd bit of the output, and so on, until finally we reach the 48th bit of the output.

2.8. Decryption Operation on DES Algorithm

Because DES uses the Feistel Structure, its encoding and decoding operations are the same only the Key schedule is reversed as shown in Figure 2.10, that means Sub-Key 16 is used in round 1 for Decryption operations instead of using the Sub-Key 16 in round 16 for Encryption Operations, in round 2 Sub-Key 15 is used, and so on. However, in decoding operations, the key generation has to create the sub keys as the series of K16, K15... K1. Figure 2.4 (Encryption Operations) and Figure 2.10 (Decryption Operations) show us the input block data to the first round of encryption operation is (the Plain-Text) of size 64-bit but the input to the first round of decryption operation is the output of the last round of encryption operation (cipher-text) because both (Initial and inverse) permutations delete each other out. That's illustrating that the data encrypted in the last round in the encryption operation becomes the data that is encrypted in the first round in the decryption operation [51].

If we look at the Figure 2.10, the function of (encryption and decryption) operation in DES algorithm is opposite to each other in round by round manner. That means round 1 in decryption operation reverses round 16 in encryption operation, round 2 in decryption operation reverses round 15 in encryption operation, and so on. That's illustrated in the following mathematical functions.

Note:

^{d} (decryption).

^{e} (encryption).

- $L_0^d = R_{16}^e$
- $R_0^d = L_{16}^e$

We can use the following mathematical equations to find another round (R_1, R_2 ……R_{16})

Finding the left half data of 32-bit

- $L_1^d = R_0^d$
- $= L_{16}^e$
- $= R_{15}^e$

Finding the left half data of 32-bit

- $R_1^d = L_0^d \oplus F(R_0^d, K_{16})$
- $= R_{16}^e \oplus F(L_{16}^e, K_{16})$
- $= [L_{15}^e \oplus F(R_{15}^e, K_{16})] \oplus F(R_{15}^e, K_{16})$
- $= L_{15}^e$

The number of rotation left bit in round key for decryption operations.

- Key not shifted in round 1.
- Key shifted one bit in rounds 2,9,16.
- Key shifted two bits in another rounds.

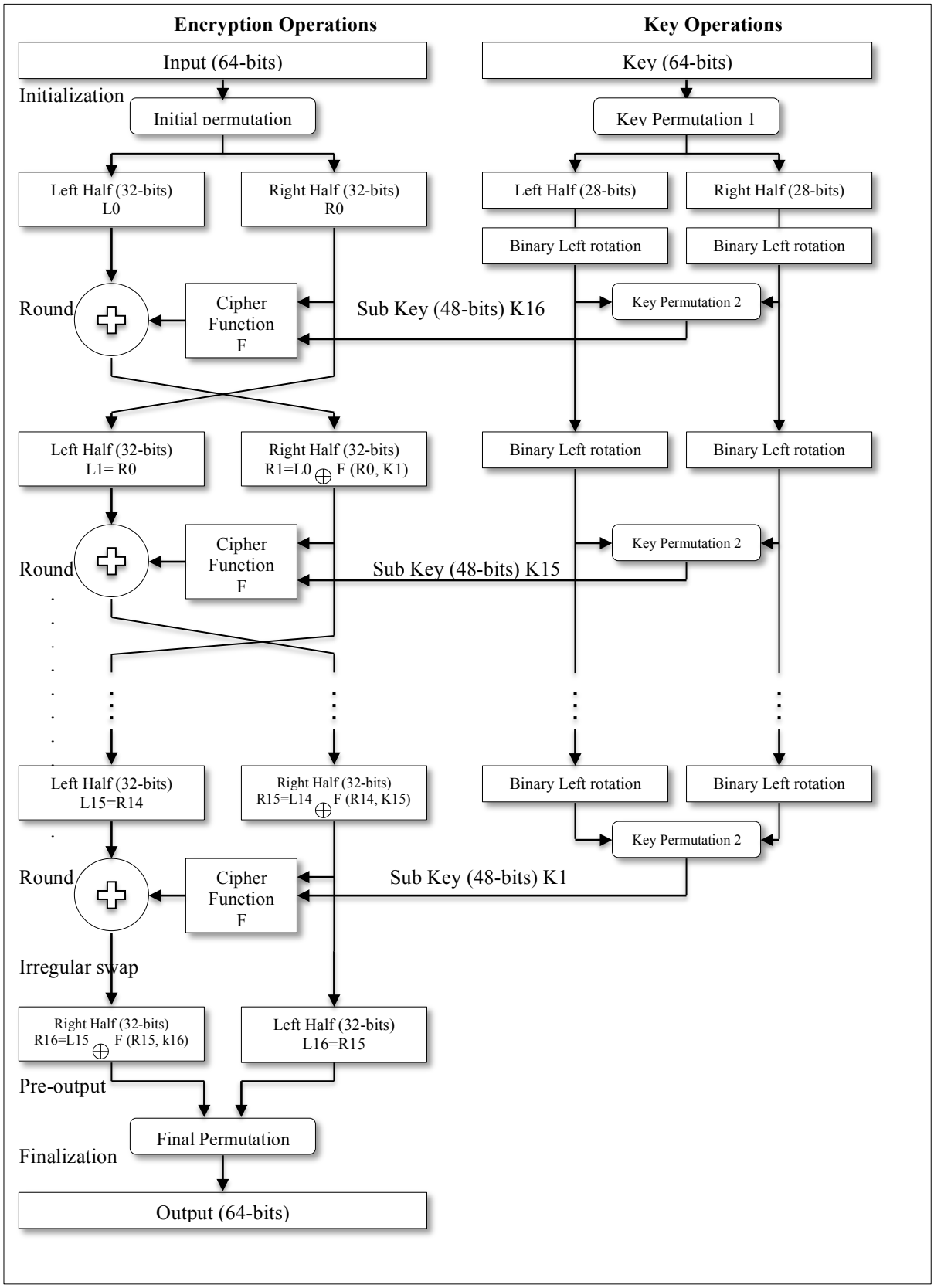

Figure 2.10. Decryption Operations

3. FAST SOFTWARE IMPLEMENTATION OF DES FOR RESOURCE CONSTRAINED DEVICE

3.1. Lightweight Cryptography

Lightweight is one of the recent keywords in cryptography, with increasing market requirements of embedded security as a background. LWC is a protocol or encryption algorithm adapted for implementation in a resource-constrained area that consists of Radio Frequently Identification tags, health-care devices, contactless smart cards, and so on. A lot of new lightweight symmetric ciphers and hash functions have been proposed, aiming at achieving low resource occupation and at the same time maintaining high level security. Lightweight cryptography is in many cases studied in the context of hardware lightweight such as low energy consumption and small circuit area, but software lightweight is also getting paid attention. Some recent researches concentrate on extensive software implementation of lightweight ciphers. Aim of LWC is to provide security requirements for resource-constrained platforms [56]. Secure applications for the Internet of Things (IoT) are constantly increasing and many of them require some lightweight cryptographic algorithms. Most LWC algorithms were not designed to be efficient in software platforms. As a result, the throughput in software of these algorithms is low on recent IoT devices. The DES is one of the algorithms that it is efficient in hardware but its design was not oriented for software platforms.

3.2. Design Approaches Lightweight Ciphers

There have been two approaches for design of lightweight ciphers [35, 36]. These approaches are:

- Optimized low-cost implementations for standardized and trusted algorithms such as DES, AES.
- Design new ciphers with the goal of having low hardware implementation costs.

Even though both approaches are valid, these approaches have some problems. The problem with the first approach is that most modern block ciphers were primarily designed with good software implementation properties in mind, and not necessarily with hardware-friendly properties. Many block cipher algorithms run in software on PCs or embedded devices. Silicon area for these platforms has become so inexpensive and hardware implementations (achieved through large chip area) are not a problem. Therefore, modern block ciphers aren't the best solution [34].

There are also problems with the second approach, designing new low cost ciphers anew. First, it is well known that it is painfully difficult to design new ciphers without security flaws.

Furthermore, it is far from straightforward to design a new cipher that has a lower hardware complexity than standard DES.

Various performance comparisons are presented in Figure 3.1, Figure 3.2, and Figure 3.3, respectively, to illustrate the rationale of the proposed design approach.

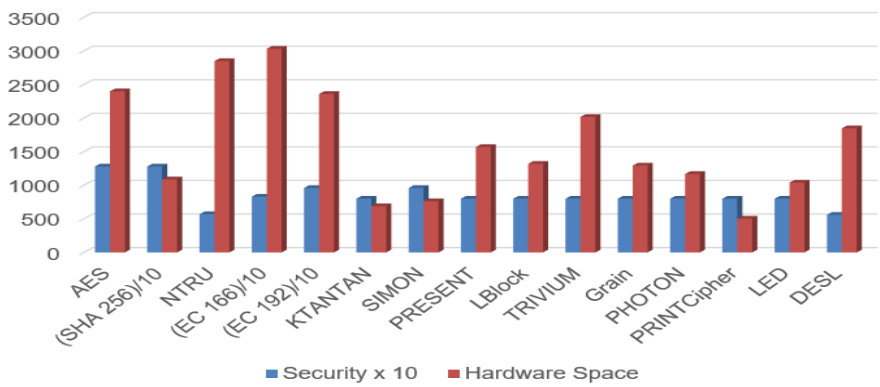

Figure 3.1. Space and security comparisons

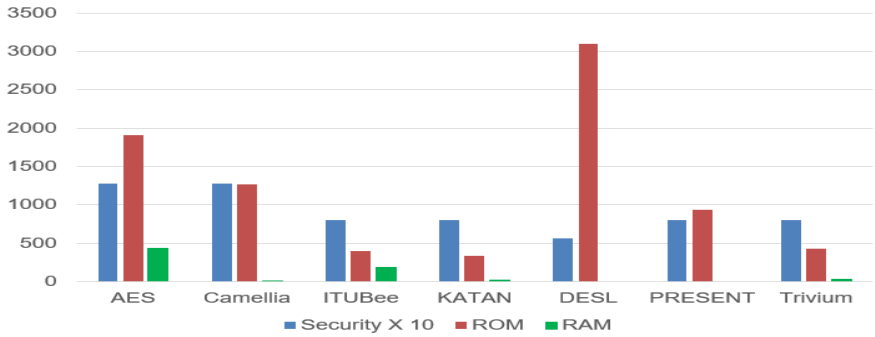

Figure 3.2. Performance comparisons for software implementation

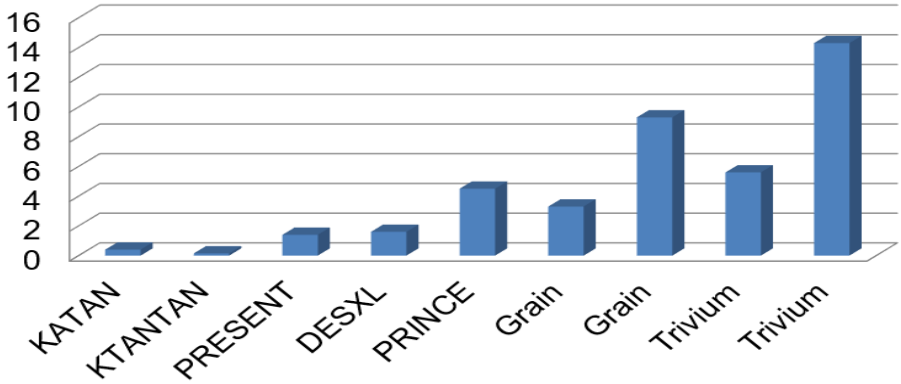

Figure 3.3. Performance comparisons for power consumption

3.3. Proposed Fast Software Implementation of DES

After we think and investigate more to find an alternative for this issue that will make DES algorithm faster, the solution that came to mind is to use new design architecture for lightweight applications. An optimum approach would be to have a well-investigated cipher. The most popular system to take into consideration the low hardware costs is Data Encryption Standard (DES) it's more efficient in hardware. DES algorithm is a hardware based block encryption algorithm. The most important reason for being effective in hardware is that the hardware costs of the 1970s were high. Feistel architecture, bit permutation and small s-boxes have been used to reduce hardware complexity. Some different techniques can be used to speed up the software implementation of the DES algorithm (Merging permutation, Realize bit permutation by table look-up, and Realize S-box substitution by table look-up.

In this thesis, we will use the first acceleration technique to speed up the DES algorithm in software. This technique is merging permutation.

The function that is the heart of the DES algorithm is shown in Figure 3.4.

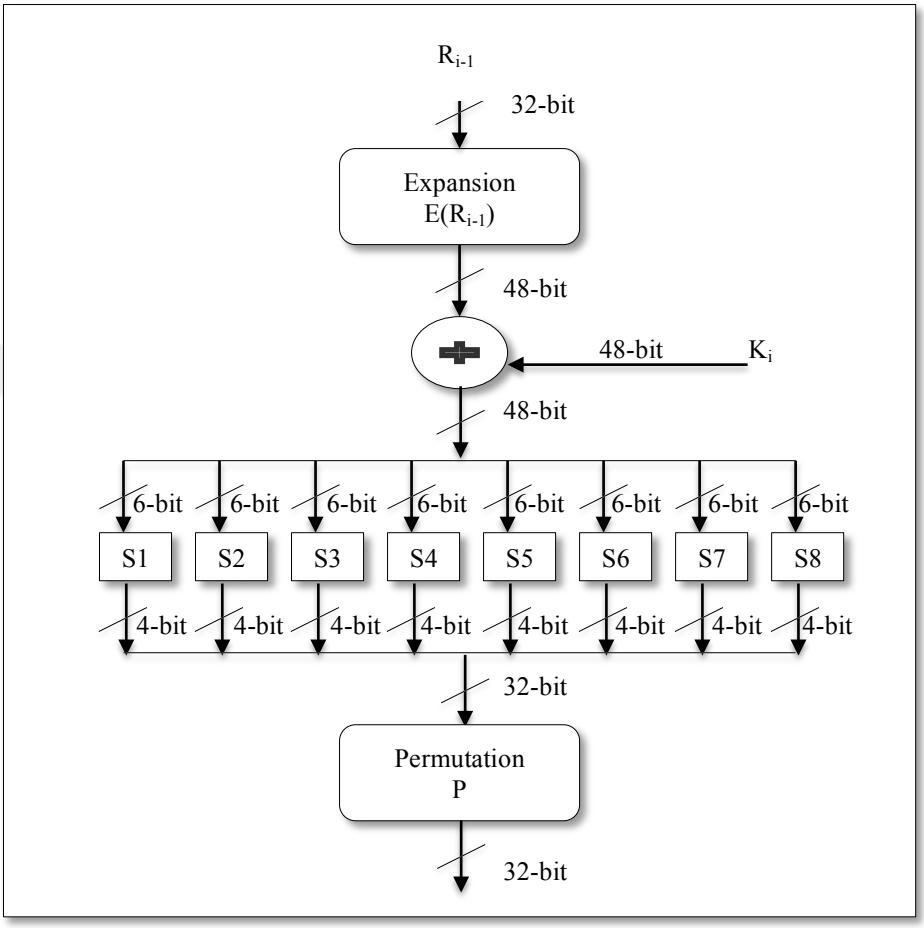

Figure 3.4. Overview of DES block cipher

In Figure 3.4 two operations of function of each round (expansion and s-box transforms) can be performed with a look-up table (subprogram calls) as show in the following code.

```
private Bitarray F(Bitarray R, Bitarray K)
{
        R = Table(E, R);
        Bitarray B = R.Xor(K);
        Bitarray S = new Bitarray(8 * 4);

        int x, y;
        Bitarray Temp;
        for (int i = 0;  i < 8; i++)
{
                x = (B[i * 6 + 0] ? 2 : 0) + (B[i * 6 + 5] ? 1 : 0);
                y = (B[i * 6 + 1] ? 8 : 0) + (B[i * 6 + 2] ? 4 : 0);
                        (B[i * 6 + 3] ? 2 : 0) + (B[i * 6 + 4] ? 1 : 0);
                Temp = new Bitarray (new byte[] {Ss[i, 16 * x + y]});
                copy(Temp, 0, S, i*4, 4);
        }

        S = Table(P, S);
        return S;
    }
```

Since, in code above the two processes (R and S) are not found in succession, the software implementation is coded as two different subprogram calls. In the code, each function (subprogram) call required stack data structure. The place where the subprogram call is made is written on top of the stack. The subprogram performs the task and code returns to main program where the subroutine was called. This process is one of the causes for slowing down the software implementation of DES algorithm. Here, a new proposed algorithm has been used to accelerate DES in software.

On the right hand of Figure 3.5 a new modified approach has been suggested to remove stack data structure problem. Because the input of both (expansion and straight permutation) is 32-bit and they are coding in different places, this will lead to stack data structure so we precede straight permutation before expansion (Merging Permutation) to avoid stack process, and after that expand the output of P of 32-bit through expansion to 48-bit, in this case for

each round we only need one permutation and one S-Box instead of two permutations and one S-Box for each round on the original algorithm, this structure is expected to achieve increasing the performance of the DES algorithm in software.

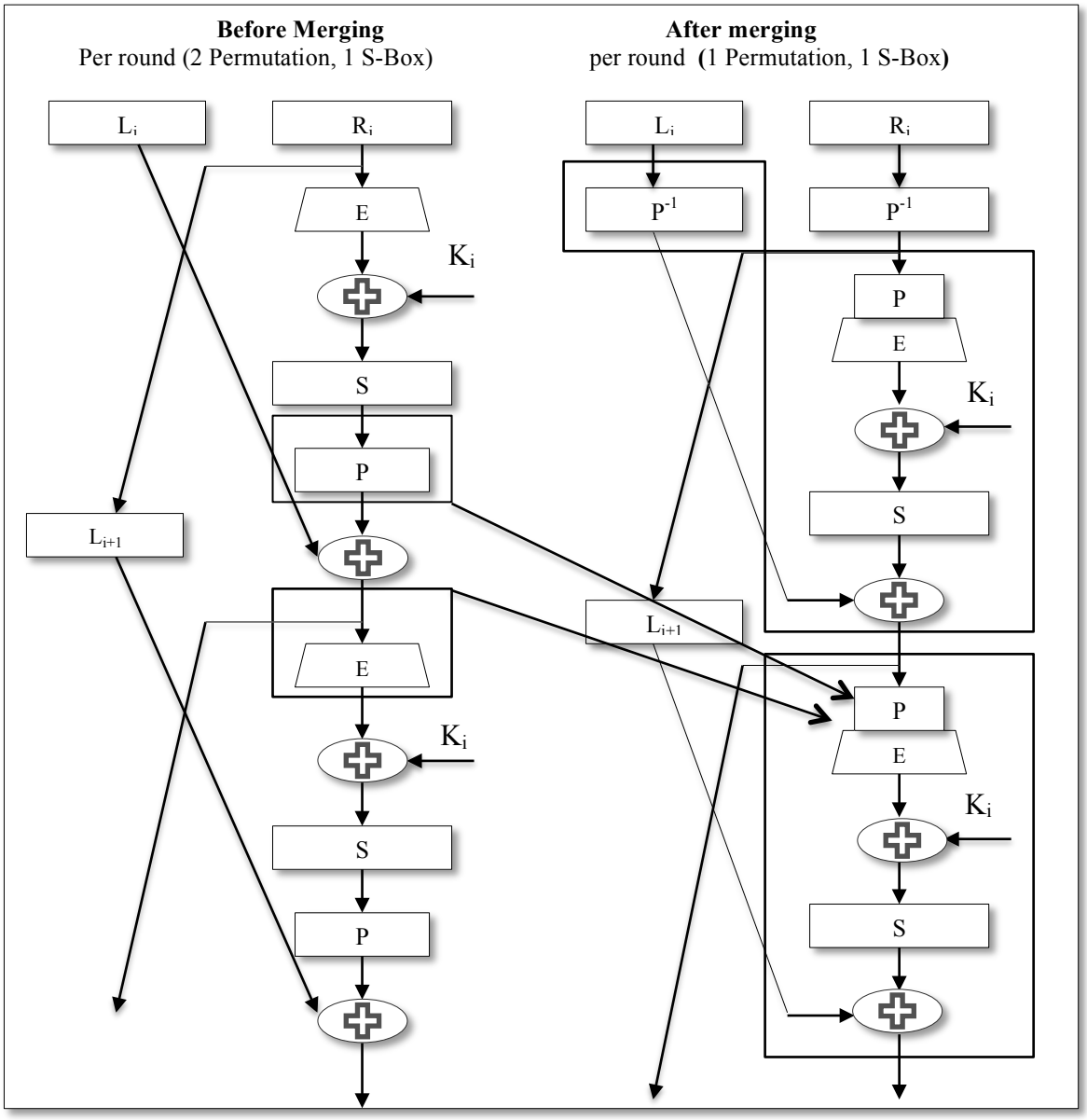

Figure 3. 5. Proposed Design Architecture

3.4. Analysis and Experiment Results of Proposed Method

After coding this new modified approach in the Python 3.0 environment, we used approximately 5 MB of txt for encryption to get efficient results from the algorithm; the same line length text is added to the file. Optimized and the old version of the algorithm is executed on the same computer. The results were recorded. The cell factor of this data is as follows.

Table 3.1. Result comparison between DES and its opponent

	Modified DES	General DES
Working Time	11,36 minutes	13,23 minutes
Number of Function Used	11.646.794	21.999.343
The Runtime of the Functions	3,27 minutes	5,16 minutes

As it can be seen in the results in Table 3.1, approximately 2 minutes time to 5 MB for the overall run time of the algorithm is optimized. If considered linearly, a 400-minute optimization result will be obtained for a 1 GB file.

3.5. Discussion

The basic requirements expected from lightweight cryptography are achieving low resource occupation (low energy consumption and small circuit area) and achieving high-level security. Many lightweight cryptography protocols are not designed to be efficient on software platforms, since designers are usually focused on hardware requirements. This section addresses this problem. We present new design architecture for improving the software implementation of the DES. The DES is a family of block ciphers. It is efficient in hardware but its design was not oriented for software platforms. The aim of this work is that find a new block cipher architecture design for lightweight applications. One-acceleration technique has been used to optimize software implementation of DES. This technique is merging permutation operations.

4. GENERATION ALTERNATIVE SBOX STRUCTURES BASED ON CHAOTIC SYSTEM

4.1. Importance of S-Box

The only nonlinear component that represented confusion characteristic for many block cipher algorithms such as DES, AES and so on is substitution box (S-box). Claude Shannon published it at the first time [41]. However, security of a cryptosystem depends on strength of s-box, weak s-box leads cryptosystem to destruction, for this reason S-box is one of the most intensive fields which has many researches on its design's criteria, which are not published yet, and how to design a new strong s-box. The DES encryption algorithm is a cryptographic protocol that has made a significant contribution to cryptography work. The DES algorithm has undergone several changes to be able to withstand diversified attacks based on technological developments. Many lightweight encryption protocols, which are becoming increasingly popular today, are also based on DES algorithm [57].

4.2. Why do we need to Generate Alternative Structures S-Box?

Many researchers have spent senior effort in analyzing DES's s-box criteria and it was designed without getting an acceptable result. For this reason the researchers are forced to search for good criteria to create good S-box for DES algorithm [1, 58]. The s-box's process on DES is explained as follows. DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. The 48-bit output from (E (Ri−1) ⊕ Ki) is separated into eight 6-bit chunks (S1-8). Each block is submitted to a single substitution function (S1-8) the output is a 4-bit chunk. The first and the last bits of the 6-input bit are combined and they define the rows in the range of 0 to 3. The 4 middle bits of the block cipher are combined which define the columns in the range of 0 to 15. When the column intersects with the row, the value in that cell will be decimal output [59].

We have been examined substitution box tables of the DES algorithm. The rationale design of DES’s s-box structures is unknown. This is an important problem in cryptology terms. The purpose of this work is to be able to generate new proposed s-box structures that

can be used as an alternative to the DES's s-box structures. There are three important contributions to the study.

- The design concept of the proposed s-box structures has been given. This removes suspicions of possible security.
- The cryptographic properties of the generated s-box structures are close to the original DES s-box structures.
- 28x28 size s-box structures are generally studied in the literature. This study has been shown that s-box structures can be designed for different sizes with random selection base.

4.3. Chaos-Based S-Box Design Studies

After studied and analyzed a set of nonlinear mathematical equations and a set of the best dynamic physical theories from many books and papers to deduce the easiest and best design for the creation of strict S-boxes with satisfying the best-cryptographic criteria. The optimization of the s-boxes is in the degree of nonlinearity value and smallest distribution table of Xor. Random selection-based methods have become increasingly popular over the last decade [23, 60]. For this reason, one of the artificial intelligence branches which is an optimal algorithm is used to achieve this purpose. A new algorithm for establishing logical design was proposed to do their job at the highest level. Our approach is called random generate and test. Random generate and test used one of the chaotic maps to get the random value and tested the value with the cryptographic criteria so as to construct the strict s-box. There are many continuous-discrete time chaotic systems that are valuable to choose as the randomness source in its highly sensitiveness. Moreover, best control parameter and initial condition are important for selection so as to avoid the overlap of trajectories generated from the initial values [61].

Many researchers resort to the chaotic system because they meet the confusion and diffusion requirements that increase the security of encryption. Of the characteristics of this system are.

- Non-periodic.
- The trajectories of the system depend on the control parameters and initial conditions.
- The system's equation is deterministic. Because the system has the nonlinearity instead of noise this generates the dynamic behavior of the system.

The process of hybridization algorithm DES with the chaos theory is useful to make this algorithm more immunize versus algebraic, differential, or linear attacks. One of the straightforward and interesting maps of chaotic maps is the logistic map. In the last few decades, some of the important and exciting information collected on this map has been detected which makes the dynamic and universality of this map very complicated.

4.4. Details of Proposed Method

The one-dimensional map discrete system (logistic map) is used in this study to generate alternative DES's S-boxes. This map demonstrates a recursive function as follows:

$$x_{i+1} = r\, x_n (1 - x_i)$$

Where $x_n \in [0, 1]$ and r is its parameter and positive number between 1 and 4. The initial condition of the logistic map has been randomly selected. 3.7 values have been used as the control parameter of the logistic map [31, 62]. After choosing the control parameter we implemented this above function with Python programming language to create an alternative DES's S-boxes as shown in Figure 4.1.

```
import random
for a in range(4):
    s=[]
    xold=random.random()

    for j in range(16):
        while True:
            xnew=3.7*xold*(1-xold)
            xold=xnew
            strValue=str(xnew)
            value=int(strValue[2:5])%16

            if value not in s:
                s.append(value)
                break

    print s
```

Figure 4.1. Codes used to generate alternative DES's S-box structures

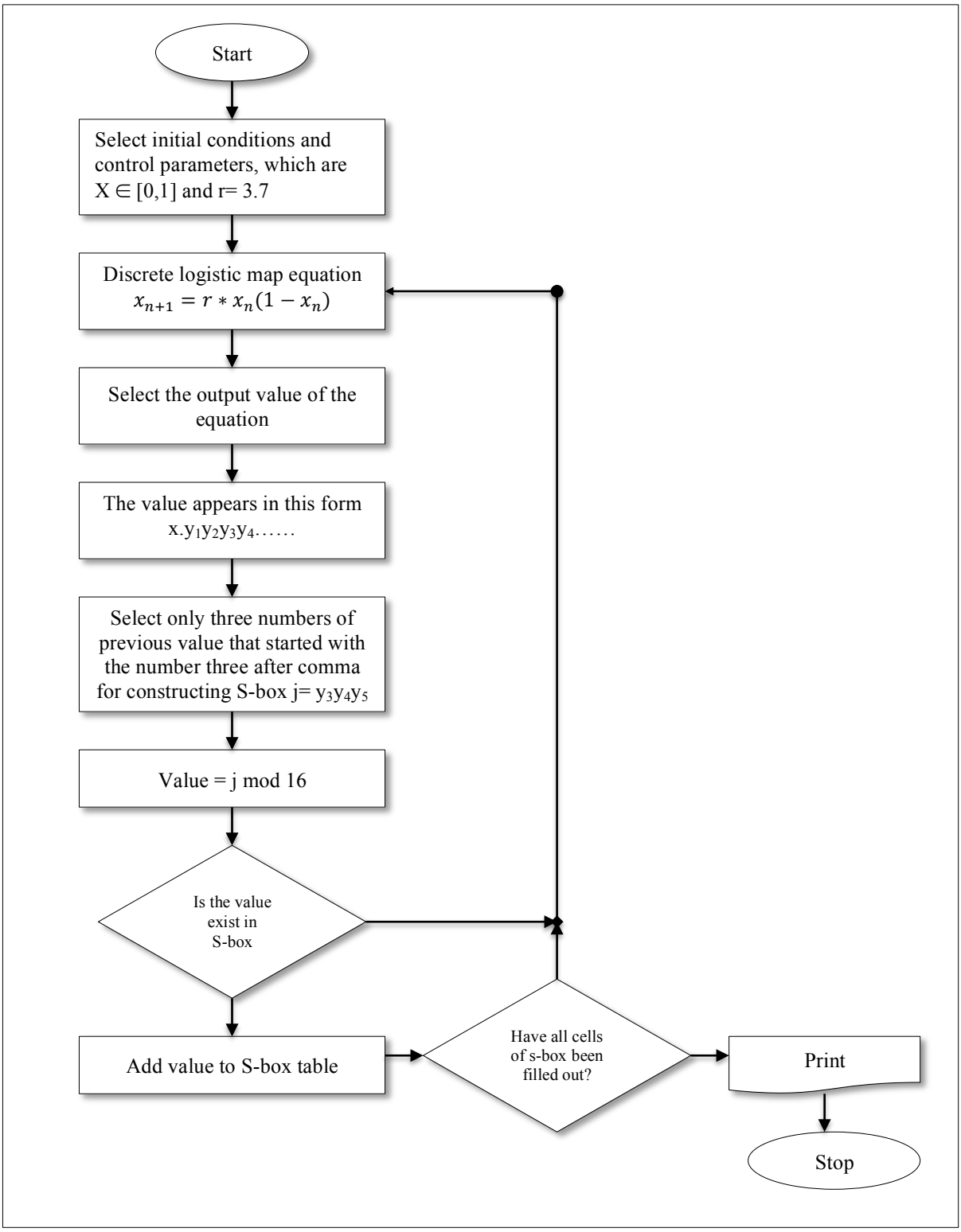

Figure 4.2. Flowchart of roadmap of constructing an alternative S-boxes

The flowchart that we designed for constructing alternative DES's 8-bit lookup table (S- Boxes) based on discrete-time logistic chaotic map consists of the following steps:

Step 1: Select the initial condition (random chosen) and control parameters (positive number) for logistic equation to get the output value (system trajectories).

Step 2: Calculating the logistic function to get fractional value.

Step 3: Convert the fractional value to the string type so as to easily selected only three digits of output value that started with number three after the comma to stay away from numerical deterioration.

Step 4: Normalizing the obtained three numbers to domain 0-15 by applying the mod 16 to get first value.

Step 5: Compare the obtained value with the S-box values if it does not exist place it into the S-box, otherwise, go to step 2 above to create a new value.

Step 6: The process will continue until all S-box's cells are filled. The alternative S-box constructed by the proposed method and the output value of using logistic map is explained in the Figure 4.1.

By an example, we will explain the operation above. For example, if the first three output values obtained from the logistic equations orbit are 0.296034 11, 1 0.740811, and 0.771073. The value between digits 3 and 6 after comma of the first output value is 603. We will mod the obtained number (603) by 16 that become (11) and placed it into the first cell of the S-box table to be the first number in s-box. The value between digits 3 and 6 after comma of the second value is 081 mod 16 = 1, so 1 becomes the second number of s-box table. The value between digits 3 and 6 after comma of the third value is 107 mod 16 = 11, but the 11 exists in the s-box table and it’s not allow the repeated value to appear in s-box table then we don’t put the number 11 into the s-box table again, and we will return to the previous step to get another value, and so on until all s-boxes cells are filled [31].

Proposed Sbox1
11, 7, 3, 1, 15, 10, 5, 8, 0, 6, 13, 2, 9, 12, 4, 14
13, 11, 9, 0, 4, 1, 8, 14, 10, 6, 7, 2, 15, 12, 5, 3
6, 5, 12, 2, 14, 13, 11, 8, 7, 15, 3, 4, 0, 1, 9, 10
2, 14, 12, 10, 15, 5, 7, 0, 13, 4, 6, 3, 8, 9, 1, 11

Proposed Sbox2
4, 8, 9, 5, 2, 12, 7, 15, 1, 10, 11, 3, 14, 6, 13, 0,
6, 7, 2, 15, 3, 11, 5, 9, 10, 1, 13, 12, 0, 14, 4, 8,
9, 14, 5, 7, 12, 3, 6, 0, 8, 1, 15, 11, 4, 2, 13, 10,
12, 14, 3, 11, 5, 2, 7, 9, 10, 4, 6, 0, 1, 13, 8, 15,

Proposed Sbox3
5, 12, 0, 1, 15, 3, 7, 11, 9, 4, 10, 2, 8, 13, 6, 14,
8, 0, 1, 2, 13, 6, 4, 12, 11, 5, 9, 7, 15, 3, 14, 10,
7, 3, 15, 8, 2, 10, 5, 12, 0, 13, 14, 4, 11, 6, 1, 9,
12, 0, 2, 15, 3, 14, 9, 11, 5, 1, 10, 7, 8, 4, 6, 13

Proposed Sbox4
14, 12, 1, 2, 15, 5, 10, 7, 8, 6, 13, 9, 4, 3, 0, 11,
6, 0, 12, 2, 5, 10, 13, 3, 1, 14, 7, 8, 15, 11, 4, 9,
4, 10, 13, 6, 0, 3, 9, 8, 5, 2, 1, 15, 11, 7, 14, 12,
11, 5, 15, 6, 7, 1, 0, 2, 4, 8, 13, 10, 12, 14, 3, 9

Proposed Sbox5
11, 9, 7, 0, 13, 2, 5, 4, 12, 10, 6, 3, 1, 14, 15, 8,
15, 8, 0, 12, 4, 2, 10, 14, 3, 11, 5, 7, 9, 6, 1, 13,
6, 2, 14, 5, 12, 10, 8, 15, 13, 7, 4, 11, 3, 0, 9, 1,
2, 8, 11, 12, 4, 5, 14, 6, 1, 10, 9, 0, 7, 13, 3, 15

Proposed Sbox6
14, 6, 10, 1, 9, 11, 4, 2, 3, 12, 15, 8, 0, 5, 13, 7,
8, 15, 12, 5, 1, 3, 7, 11, 10, 9, 14, 0, 6, 2, 13, 4,
9, 12, 6, 15, 11, 10, 13, 7, 5, 2, 3, 14, 1, 0, 8, 4,
3, 12, 7, 14, 1, 9, 11, 6, 0, 10, 4, 13, 5, 15, 8, 2

Proposed Sbox7
6, 4, 3, 8, 5, 7, 15, 12, 1, 14, 10, 2, 9, 11, 0, 13,
1, 3, 10, 8, 14, 2, 4, 11, 5, 13, 9, 7, 6, 12, 15, 0,
4, 2, 10, 13, 14, 7, 1, 8, 9, 11, 6, 15, 12, 0, 3, 5,
0, 9, 13, 6, 3, 11, 4, 7, 5, 8, 12, 15, 1, 2, 10, 14

Proposed Sbox8
9, 3, 13, 8, 10, 15, 5, 7, 12, 0, 2, 14, 11, 1, 6, 4
1, 3, 12, 10, 6, 2, 9, 11, 0, 14, 7, 15, 4, 5, 8, 13
3, 4, 9, 12, 6, 14, 1, 8, 0, 2, 5, 13, 10, 15, 11, 7
10, 14, 5, 13, 7, 1, 15, 11, 12, 8, 3, 2, 0, 9, 6, 4

Figure 4.3. Generated Proposed S-box structures

4.5. Analysis and Test Results

Nonlinearity, balance, resistance to differential and linear attack play an important role in the performance evaluation of s-box design [63]. These features of s-box structures used in the original DES algorithm are examined. The obtained results are given in Table 4.1. The calculated values of the s-box structures listed in Figure 4.3 these five criteria are given in Table 4.2.

Table 4.1. Performance Value for classical DES S-box Structure

S-box	Index	Nonlinearity	Balance	XOR Table	LAT
DES Sbox1	1	18	True	16	12
	2	22	True		
	3	20	True		
	4	18	True		
DES Sbox2	1	18	True	16	10
	2	18	True		
	3	20	True		
	4	22	True		
DES Sbox3	1	18	True	16	16
	2	20	True		
	3	22	True		
	4	18	True		
DES Sbox4	1	22	True	16	16
	2	22	True		
	3	22	True		
	4	22	True		
DES Sbox5	1	20	True	16	14
	2	18	True		
	3	20	True		
	4	22	True		
DES Sbox6	1	20	True	16	14
	2	20	True		
	3	20	True		
	4	20	True		
DES Sbox7	1	20	True	16	14
	2	14	True		
	3	22	True		
	4	18	True		
DES Sbox8	1	22	True	16	16
	2	20	True		
	3	20	True		
	4	22	True		

Table 4.2. Performance Value of proposed S-box Structure

S-box	Index	Nonlinearity	Balance	XOR Table	LAT
Proposed Sbox1	1	24	True	16	14
	2	20	True		
	3	22	True		
	4	22	True		
Proposed Sbox2	1	24	True	16	14
	2	20	True		
	3	22	True		
	4	22	True		
Proposed Sbox3	1	20	True	14	14
	2	22	True		
	3	20	True		
	4	22	True		
Proposed Sbox4	1	22	True	16	14
	2	24	True		
	3	20	True		
	4	20	True		
Proposed Sbox5	1	20	True	18	12
	2	20	True		
	3	16	True		
	4	20	True		
Proposed Sbox6	1	22	True	14	14
	2	22	True		
	3	22	True		
	4	22	True		
Proposed Sbox7	1	22	True	18	12
	2	22	True		
	3	22	True		
	4	22	True		
Proposed Sbox8	1	18	True	16	16
	2	20	True		
	3	20	True		
	4	20	True		

4.6. Discussion

In this section, substitution box tables of the DES algorithm have been examined. The design rationale of DES s-box structures is unknown. This is an important problem in cryptology terms. The purpose of this section is to be able to generate s-box structures that can be used as an alternative to the DES algorithm. There are three important contributions to this section.

- The design concept of the proposed s-box structures has been given. This removes suspicions of possible security.
- The cryptographic properties of the generated s-box structures are close to the original DES s-box structures.
- 2^8x2^8 size s-box structures are generally studied in the literature. This section has been shown that s-box structures can be designed for different sizes with random selection base.

5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

This thesis is dealing with two outputs, the first one is how to accelerate DES in software, and the second one is how to clear the s-box mathematically.

First: In the digitized world, information security is becoming increasingly important. Researchers are working on new encryption techniques to ensure information security criteria such as privacy, integrity, and authenticity. In the last decades, lightweight cryptography has become one of the most important topics to design new encryption systems for source limited devices. In this study, one different acceleration technique have been used to speed up the DES algorithm in software. This technique has not been used previously in the literature, and is the unique aspect of the thesis work.

Second: Cryptographic structures which can be alternative to DES s-box structures have been produced. Generated s-box structures are based on a random selection approach. Chaotic systems have been used as a source of randomness. In the study, the first eight s-boxes produced by the proposed method are listed. The obtained performance values are closer to the original DES algorithm. If more s-box structures are generated with the proposed algorithm, then the s-box structures can be produced with better performance criteria.

5.2. Future Work

Future studies aim to reach the following objectives:

- The changes made to the architecture of the DES algorithm will also be tried to be adapted for other encryption algorithms.
- We will try to reduce the power consumption of the algorithms by optimizing the logic elements used at hardware level.
- The most appropriate data structures will be selected to improve the software implementation performance of the algorithms.
- Lightweight architects will be searched for stream cipher systems as well as block cipher systems.

REFERENCES

[1] **Paar, I.C. and Pelzl, I.J.,** 2010. The Data Encryption Standard (DES) and alternatives, *In Understanding Cryptography, Springer Berlin Heidelberg*, p. 55-86.

[2] **Menezes, A.J., Van Oorschot, P.C. and Vanstone, S.A.,** 1996. DES. *In Handbook of applied cryptography*, *CRC press*, p. 250-259.

[3] **Schneier, B.,** 2007. Data Encryption Standard (DES), *In Applied cryptography: protocols, algorithms, and source code in C, john Wiley & sons*.

[4] **Stinson, D.R.,** 2005. The data encryption standard, In *Cryptography: theory and practice*, CRC press, p.95-100.

[5] **Zakaria, N.H., Mahmod, R., Udzir, N.I. and ZUKARNAIN, Z.A.,** 2015. ENHANCING ADVANCED ENCRYPTION STANDARD (AES) S-BOX GENERATION USING AFFINE TRANSFORMATION, *Journal of Theoretical & Applied Information Technology*, ***72***, p.1.

[6] **Knudsen, L.R. and Robshaw, M.,** 2011. DES, in the block cipher companion, *Springer Science & Business Media*, p.13-34.

[7] **Martin, K.M.,** 2012. Block Cipher, *in Everyday Cryptography Fundemental Principles and Applications (2012)*, p. 114-123.

[8] **Smid, M.E. and Branstad, D.K.,** 1988. Data encryption standard: past and future, *Proceedings of the IEEE*, **76**, p. 550-559.

[9] **Alanazi, H., Zaidan, B.B., Zaidan, A.A., Jalab, H.A., Shabbir, M. and Al-Nabhani, Y.,** 2010. New comparative study between DES, 3DES and AES within nine factors, *arXiv preprint arXiv:1003.4085,* **2**.

[10] **Singh, A.K. and Varshney, S.,** 2014. Enhanced Data Encryption Standard using Variable Size Key (128N Bits) and 96 Bit Subkey, *International Journal of Computer Applications*, ***98***.

[11] **Kaul, V. Narayankhedkar, S K. Khaitan, K. Patil, A. and Dube, R.,** 2013. Enhanced Data Encryption Algorithm for Next Generation Networks, *IJAIS Proceedings on International Conference and workshop on Advanced Computing 2013 ICWAC*, **4**, p. 8-17.

[12] **Khan, M.N., Wahid, I. and Ikram, A.A.,** 2012. The FastDES: A New Look of Data Encryption Standard, *International Journal of Computer Applications*, ***39***.

[13] **Sudha, S., Viswanatham, V.M., Brindha, K., Agilandeeswari, L. and Ramya, G.,** 2012. Implementation of Enhanced Data Encryption Standard on MANET with less energy consumption through limited computation, *International Journal of Engineering Research and Development eISSN*, **2**, p. 46-52.

[14] **Al-Hamami, A., Al-Hamami, M.A. and Hashem, S.H.,** 2011. A proposed modified data encryption standard algorithm by using fusing data technique, *World of Computer Science and Information Technology Journal* (*WCSIT)*, **1**, p. 88-91.

[15] **Hashem, S.H. and Mohammad, A., 2011. AL-Hamami, Alaa H. AL-Hamami**, Developing a Block-Cipher-Key Generator Using Philosophy of Data Fusion Technique, *Journal of Emerging Trends in Computing and Information Sciences*, **2**, p. 222-227.

[16] **Mohammed, A.E. and Abdalla, F.M.,** 2016. DES Security Enhancement using Genetic Algorithm, SUST Journal of Engineering and Computer Science (JECS), **17**.

[17] **Arrag, S., Hamdoun, A., Tragha, A. and Khamlich, S.E.,** 2013. Replace AES key expansion algorithm by modified genetic algorithm, *Applied Mathematical Sciences*, **7**, p.7161-7171.

[18] **Verma, J. and Prasad, S., 2009.** Security Enhancement in Data Encryption Standard. In *International Conference on Information Systems, Technology and Management*, *Springer, Berlin, Heidelberg,* p. 325-334.

[19] **Han, S.J., Oh, H.S. and Park, J.,** 1996, September. The improved data encryption standard (DES) algorithm. In *Spread Spectrum Techniques and Applications Proceedings, IEEE 4th International Symposium on*, **3**, p. 1310-1314.

[20] **Zakaria, N.H., Mahmod, R., Udzir, N.I. and ZUKARNAIN, Z.A.,** 2015. ENHANCING ADVANCED ENCRYPTION STANDARD (AES) S-BOX GENERATION USING AFFINE TRANSFORMATION. *Journal of Theoretical & Applied Information Technology*, **72**.

[21] **Leander, G., Paar, C., Poschmann, A. and Schramm, K.,** 2007. New lightweight DES variants. In *International Workshop on Fast Software Encryption, Springer, Berlin, Heidelberg,* p. 196-210.

[22] **Seberry, J., Zhang, X.M. and Zheng, Y.,** 1993. Systematic generation of cryptographically robust S-boxes. In *Proceedings of the 1st ACM Conference on Computer and Communications Security, ACM*, p. 171-182.

[23] **Detombe, J. and Tavares, S.,** 1992. Constructing large cryptographically strong S-boxes. In *International Workshop on the Theory and Application of Cryptographic Techniques, Springer, Berlin, Heidelberg,* p. 165-181.

[24] **Jakimoski, G. and Kocarev, L.,** 2001. Chaos and cryptography: block encryption ciphers based on chaotic maps. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **48**, pp.163-169.

[25] **Özkaynak, F. and Özer, A.B.,** 2010. A method for designing strong S-Boxes based on chaotic Lorenz system. *Physics Letters A*, **374**, pp. 3733-3738.

[26] **Özkaynak, F., Çelik, V. and Özer, A.B.,** 2017. A new S-box construction method based on the fractional-order chaotic Chen system. *Signal, Image and Video Processing*, **11**, p. 659-664.

[27] **Çavuşoğlu, Ü., Zengin, A., Pehlivan, I. and Kaçar, S.,** 2017. A novel approach for strong S-Box generation algorithm design based on chaotic scaled Zhongtang system. *Nonlinear Dynamics*, **87**, p. 1081-1094.

[28] **Liu, G., Yang, W., Liu, W. and Dai, Y.,** 2015. Designing S-boxes based on 3-D four-wing autonomous chaotic system. *Nonlinear Dynamics*, **82**, p. 1867-1877.

[29] **Belazi, A., Khan, M., El-Latif, A.A.A. and Belghith, S.,** 2017. Efficient cryptosystem approaches: S-boxes and permutation–substitution-based encryption. *Nonlinear Dynamics*, **87**, p. 337-361.

[30] **Lambić, D.,** 2018. S-box design method based on improved one-dimensional discrete

Chaotic map. *Journal of Information and Telecommunication*, p.1-11.

[31] **Özkaynak, F.,** 2017. Construction of robust substitution boxes based on chaotic systems. *Neural Computing and Applications*, p. 1-10.

[32] **Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J. and Wingers**, L., 2015. The SIMON and SPECK lightweight block ciphers. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, p. 1-6.

[33] **Katagi, M. and Moriai, S.,** 2008. Lightweight cryptography for the Internet of things, *Sony Corporation*, p.7-10.

[34] **Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y. and Vikkelsoe, C.,** 2007. PRESENT: An ultra-lightweight block cipher, In *International Workshop on Cryptographic Hardware and Embedded Systems, Springer, Berlin, Heidelberg*, p. 450-466.

[35] **Davio, M., Desmedt, Y., Goubert, J., Hoornaert, F. and Quisquater, J.J.,** 1984. Efficient hardware and software implementations for the DES, In *Workshop on the Theory and Application of Cryptographic Techniques, Springer, Berlin, Heidelberg, p. 144-146.*

[36] **Verbauwhede, I., Hoornaert, F., Vandewalle, J. and De Man, H.J.,** 1988. Security and performance optimization of a new DES data encryption chip, *IEEE Journal of Solid-State Circuits*, **23**, p. 647-656.

[37] **Mahajan, P. and Sachdeva, A.,** 2013. A study of encryption algorithms AES, DES and RSA for security, *Global Journal of Computer Science and Technology*.

[38] **Paar. C, Aachen, R. and Germany.,** 2013. Cryptographic Engineering, in *Introduction to Block Ciphers: DES and AES and Lightweight Block Ciphers for RFIDs*, MEAD Education S.A, p. 2-32, 58-68.

[39] **Stallings, W.,** 2005. The data encryption standard, in Cryptography and network security: principles and practices, *Prentice Hall*, p. 62-90.

[40] **Landau, S.,** 2000. Standing the test of time: The data encryption standard, *Notices of AMS*, **47**, p. 341-349.

[41] **Shannon, C.E.,** 1949. Communication theory of secrecy systems. *Bell Labs Technical*

Journal, **4**, p. 656-715.

[42] **McAndrew, A.,** 2016. Block ciphers and the data encryption standard, *in Introduction to Cryptography with open-source software*, *CRC Press*, p.167-214.

[43] **Karthik, S. and Muruganandam, A.,** 2014. Data Encryption and Decryption by using Triple DES and performance analysis of crypto system, *International Journal of Scientific Engineering and Research*, **2**, p. 24-31.

[44] **Rahmati, A., Qian, A. and Zhong, L.,** 2007. Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, ACM, p. 265-272. .

[45] **Damaševičius, R., Štuikys, V. and Toldinas, J.,** 2013. Methods for measurement of energy consumption in mobile devices, *Metrology and measurement systems*, **20**, p.419-430.

[46] **Kaps, J.P. and Paar, C.,** 1998. Fast DES implementations for FPGAs and its application to a universal key-search machine. In *International Workshop on Selected Areas in Cryptography, Springer, Berlin, Heidelberg,* p. 234-247.

[47] **Delfs, H., Knebl, H. and Knebl, H.,** 2007. Symetric-Key Encryption, in *Introduction to cryptography, Springer, Verlag Berlin Heidelberg*, **2**, p. 11-25.

[48] **FIPS, P.,** 1999. 46-3: Data encryption standard (des), in *National Institute of Standards and Technology*, **25**, pp.1-22.

[49] **Wobst, R.,** 2007. Feistel Networks, in Cryptology unlocked, John Wiley & Sons, p. 133-162.

[50] **PUB, F.,** 1999. Data Encryption Standard (DES), *FIPS PUB*, p. 46-3.

[51] **Yang, B., Wu, K. and Karri, R.,** 2004. Scan based side channel attack on dedicated hardware implementations of data encryption standard, In *Test Conference, 2004 Proceedings, ITC 2004, International*, IEEE, p. 339-344.

[52] **Smid, M.E. and Branstad, D.K.,** 1988. Data encryption standard: past and future, *Proceedings of the IEEE*, **76**, p. 550-559.

[53] **Feldmeier, D.C.,** 1989. A high-speed software DES implementation, *Computer Communication Research Group, Bellcore*.

[54] **Richard, A.,** 2001. DES and AES, in An Introduction to Cryptography, *CRC 2007*, P. 31-51.

[55] **Ehrsam, W.F., Matyas, S.M., Meyer, C.H. and Tuchman, W.L.,** 1978. A cryptographic key management scheme for implementing the Data Encryption Standard, *IBM Systems Journal*, **17**, p. 106-125.

[56] **Katagi, M. and Moriai, S.,** 2008. Lightweight cryptography for the Internet of things, *Sony Corporation*, p. 7-10.

[57] **Zaibi, G., Kachouri, A., Peyrard, F. and Fournier-Prunaret, D.,** 2009. On dynamic chaotic S-Box, In *Information Infrastructure Symposium, 2009, GIIS'09, Global,* IEEE, p. 1-5.

[58] **Adams, C. and Tavares, S.,** 1989. Good S-boxes are easy to find, In *Conference on the Theory and Application of Cryptology, Springer, New York, NY,* p. 612-615.

[59] **Gargiulo, J.,** 2002. S-Box Modifications and Their Effect in DES-like Encryption Systems, *SANS Institute InfoSec Reading Room*.

[60] **Özkaynak, F.,** 2017. From Biometric Data to Cryptographic Primitives: A New Method for Generation of Substitution Boxes, In *Proceedings of the 2017 International Conference on Biomedical Engineering and Bioinformatics, ACM,* p. 27-33.

[61] **Özkaynak, F.,** 2015. A novel method to improve the performance of chaos based evolutionary algorithms, *Optik-International Journal for Light and Electron Optics*, **126**, p. 5434-5438.

[62] **Borujeni, S.E. and Ehsani, M.S.,** 2015. Modified logistic maps for cryptographic application, *Applied Mathematics*, **6**, p. 773.

[63] **Cusick, T.W. and Stanica, P.,** 2017. Data Encryption Standard, in *Cryptographic Boolean functions and applications*, Academic Press, p. 196-202.

CURRICULUM VITAE

Mukhlis Ibrahem Muhamad Sharef

E-mail: mukhlisperavdaly85@gmail.com

Telephone: +9647504030604

Nationality : Iraq

Place of birth : Duhok

Date of birth : 16 / 7 / 1984

Marital status : Married

EDUCATION

2016 – 2018 MCE (Master of Computer Engineering), Firat University, Elazig, Turkey.

2004 – 2007 Bachelor of Computer Science, Duhok University, Iraq.

2001 – 2003 High School, Sheladze High School, Duhok, Iraq.

1998 – 2000 Intermediate School, Sheladze intermediate School, Duhok, Iraq

1990 – 1997 Primary School, Koran Primary School, Duhok, Iraq

Work experiences:

- Teacher of Computer Science at Sheladze High School, teaching (Database). From 2008-2018.
- Have worked as team leader with Election Sun Network, from 2004-2006.
- Have 13 years' experience in **repairing Electronic Devices and have worked as an Electrician.**

International Conference Papers

- Certificate of Presentation and Participation in the **International Symposium on Digital Forensic and Security (ISDFS 2018)**, Antalya, Turkey on March 22-25-2018.
- Certificate of Presentation and Participation in the **International Artificial Intelligence and Data Processing symposium (IDAP'17)**, Malatya, Turkey on 27 – 7 - 2017.
- Certificate of participating in the **International Cyber Security Workshop Certificate Program**, Gelişim University/Istanbul-Turkey, on May 23-27, 2016.

Languages: Kurdish, English, and Arabic.