Analysis of DES Algorithm

Project Report Submitted in Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Engineering
in
Information Technology

Submitted by

Bhavya Sen: (Roll No.19UITE9023)

Jeet Agnihotri: (Roll No.19UITE9024)

Under the Supervision of

Simran Choudhary
Assistant Professor

Department of Computer Science and Engineering
MBM University, Jodhpur

June, 2022

This page was intentionally left blank.

Analysis of DES Algorithm

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering
in
Information Technology

Submitted by

Bhavya Sen: (Roll No.19UITE9023)

Jeet Agnihotri: (Roll No.19UITE9024)

Under the Supervision of

Simran Choudhary
Assistant Professor

Department of Computer Science and Engineering
MBM University, Jodhpur

June, 2022

This page was intentionally left blank.

Department of Computer Science & Engineering

M.B.M. University
Ratanada, Jodhpur, Rajasthan, India –342011

CERTIFICATE

This is to certify that the work contained in this report entitled **"Analysis of DES Algorithm**" is submitted by the group members Ms. Bhavya Sen (Roll. No: 19UITE9023), Ms. Jeet Agnihotri (Roll No: 19UITE9024) to the Department of Computer Science & Engineering, M.B.M. University, Jodhpur, for the partial fulfilment of the requirements for the degree of **Bachelor of Engineering** in **Information Technology**.

They have carried out their work under my supervision. This work has not been submitted else-where for the award of any other degree or diploma.

The project work in our opinion, has reached the standard fulfilling of the requirements for the degree of Bachelor of Engineering in Information Technology in accordance with the regulations of the Institute.

Ms. Simran Choudhary
Assistant Professor
(Supervisor)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

Dr. Nemi Chand Barwar
(Head)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

This page was intentionally left blank.

DECLARATION

I, ***Jeet Agnihotri,*** hereby declare that this major project titled **"Analysis of DES Algorithm"** is a record of original work done by me under the mentorship of ***Dr. Shrwan Ram*** and guidance of ***Ms. Simran Choudhary.***

I, further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

Jeet Agnihotri
VIIIth Semester, IT
Enroll. – 18R/04611
Roll No. – 19UTIE9024

This page was intentionally left blank.

ACKNOWLEDGEMENT

First of all, I am indebted to the GOD ALMIGHTY for giving me an opportunity to excel in my efforts to complete this project on time.

I am extremely grateful to **Dr. N.C. Barwar**, Head of Department, Department of Computer Science & Engineering, MBM University, Jodhpur, for providing all the required resources for the successful completion of my project.

I wish to express my deep sense of gratitude to my project guide ***Ms. Simran Choudhary***, Assistant Professor, for her able guidance and useful suggestions, which helped me in completing the project, on time.

I will be failing in duty if I do not acknowledge with grateful thanks to the authors of the references and other literatures referred to in this seminar.

Last but not the least; I am very much thankful to my parents who guided me in every step which I took.

This page was intentionally left blank.

ABSTRACT

DES is a symmetric block cipher (shared secret key), with a key length of 56-bits. Published as the Federal Information Processing Standards (FIPS) 46 standard in 1977. After the publication of DES, it was criticized severely for the small key length, which could make the cipher vulnerable to brute-force attack. DES was officially withdrawn in 2005. For a modern-day computer, a brute force attack using 2^{56} keys is a matter of few days. This project studies the DES algorithm by increasing the key size in order to try to overcome small key size problem. DES key generation algorithm is modified in this project to use 128-bit keys.

DES is first implemented in Sage Math notebooks using Sagemath DES library to build an understanding of DES. Next step was to code DES algorithm. A modified DES is implemented after changing key scheduling algorithm and increasing the key size to 128 bits. After implementation, analysis is performed by noting differences between performance of standard DES and modified DES. The analysis is performed by measuring avalanche effect by implementing both algorithms on same plaintext.

This page was intentionally left blank.

Contents

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

List of Tables

Chapter 1

INTRODUCTION

We are living in the information age. We need to keep information about every aspect of our lives. In other words, information is an asset that has a value like any other asset. As an asset, information needs to be secured from attacks. To be secured, information needs to be hidden from unauthorized access (confidentiality), protected from unauthorized change (integrity), and available to an authorized entity when it is needed (availability). There are three security goals:

1. Confidentiality
2. Integrity
3. Availability

Until a few decades ago, the information collected by an organization was stored on physical files. The confidentiality of the files was achieved by restricting the access to a few authorized and trusted people in the organization. In the same way, only a few authorized people were allowed to change the contents of the files. Availability was achieved by designating at least one person who would have access to the files at all times. With the advent of computers, information storage became electronic. Instead of being stored on physical media, it was stored in computers. The three security requirements, however, did not change. The files stored in computers require confidentiality, integrity, and availability. The implementation of these requirements, however, is different and more challenging. During the last two decades, computer networks created a revolution in the use of information. Information is now distributed. Authorized people can send and retrieve information from a distance using computer networks. Although the three above-mentioned requirements: confidentiality, integrity, and availability have not changed, they now have some new dimensions. Not only should information be confidential when it is stored in a computer; there should also be a way to maintain its confidentiality when it is transmitted from one computer to another. Our three goals of security confidentiality,

integrity, and availability can be threatened by security attacks. The implementation of security goals needs some techniques. Two techniques are prevalent today: one is very cryptography and one is steganography. This project studies DES algorithm of cryptography.

1.1 Cryptography

Cryptography provides for secure communication in the presence of malicious third-parties—known as adversaries. Encryption uses an algorithm and a key to transform an input (i.e., plaintext) into an encrypted output (i.e., ciphertext). A given algorithm will always transform the same plaintext into the same ciphertext if the same key is used. Algorithms are considered secure if an attacker cannot determine any properties of the plaintext or key, given the ciphertext. An attacker should not be able to determine anything about a key given a large number of plaintext/ciphertext combinations which used the key.

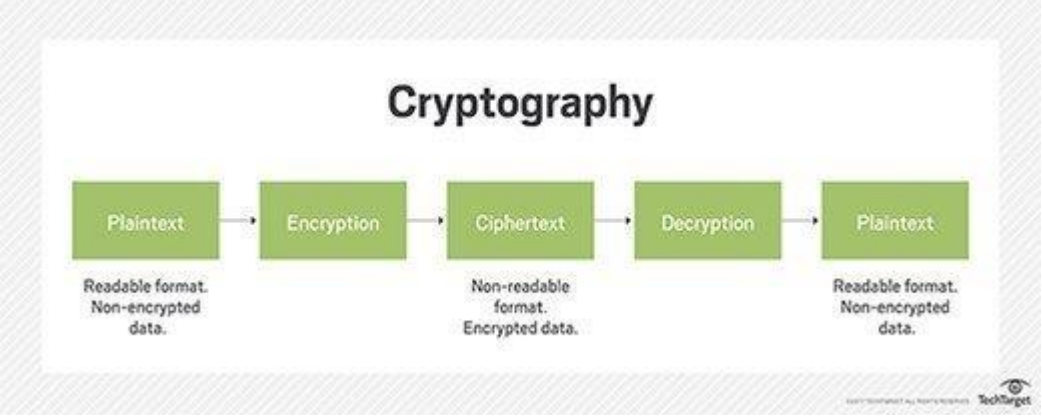

Figure 1.1 : Cryptography [2]

Principles of Cryptography

The most important principle to keep in mind is that we should never attempt to design our own cryptosystem. The world's most brilliant cryptographers (including Phil Zimmerman and Ron Rivest) routinely create cryptosystems with serious security flaws in them. In order for a cryptosystem to be deemed "secure," it must face intense scrutiny from the security community. Never rely on security through obscurity, or the fact that

attackers may not have knowledge of our system. Remember that malicious insiders and determined attackers will attempt to attack our system.

The only things that should be “secret” when it comes to a secure cryptosystem are the keys themselves. We should be sure to take appropriate steps to protect any keys that our systems use. Never store encryption keys in clear text along with the data that they protect. This is akin to locking our front door and placing the key under the doormat. It is the first place an attacker will look. Here are three common methods for protecting keys (from least secure to most secure):

1. Store keys in a filesystem and protect them with strong access control lists (ACLs). Remember to adhere to the principal of least privilege.
2. Encrypt our data encryption keys (DEKs) with a second key encrypting key (KEK). The KEK should be generated using password-based encryption (PBE). A password known to a minimal number of administrators can be used to generate a key using an algorithm such as bcrypt, scrypt, or PBKDF2 and used to bootstrap the cryptosystem. This removes the need to ever store the key unencrypted anywhere.
3. A hardware security module (HSM) is a tamper-resistant hardware appliance that can be used to store keys securely. Code can make API calls to an HSM to provide keys when needed or to perform decryption of data on the HSM itself.

1.2 Types of Cryptography

1. Symmetric Key Encryption

Encryption is a process to change the form of any message in order to protect it from reading by anyone. In Symmetric-key encryption the message is encrypted by using a key and the same key is used to decrypt the message which makes it easy to use but less secure. It also requires a safe method to transfer the key from one party to another. Examples of symmetric key cryptography include AES, DES, and 3DES. Key exchange protocols used to establish a shared encryption key include Diffie-Hellman (DH), elliptic curve (EC) and RSA.

Data Encryption Standard

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline. For a modern-day computer, a brute force attack using 2^{56} keys is an easy nut to crack. We try to address this problem in this study.

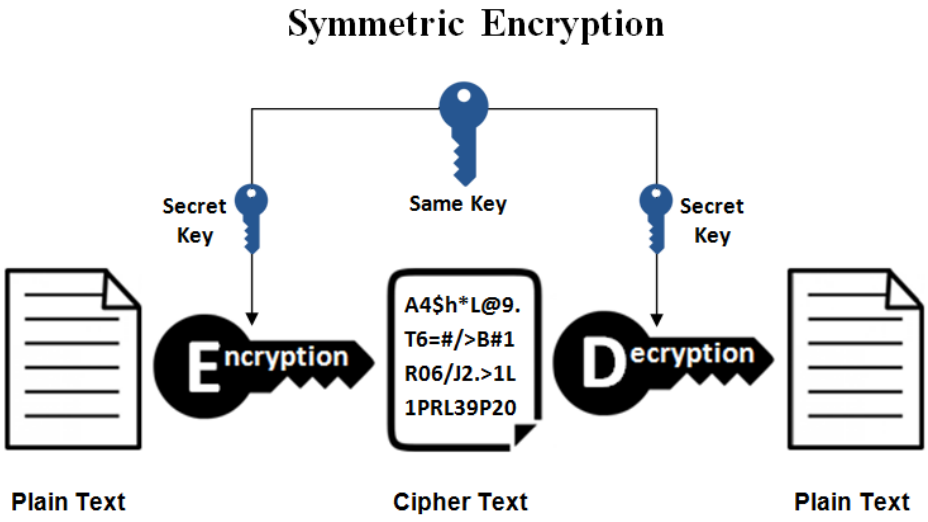

Figure 1.2 : Symmetric Encryption [3]

2. Asymmetric Key Encryption

Asymmetric Key Encryption is based on public and private key encryption technique. It uses two different keys to encrypt and decrypt the message. It is

more secure than symmetric key encryption technique but is much slower. A public key is a cryptographic key that can be used by any person to encrypt a message so that it can only be decrypted by the intended recipient with their private key. A private key -also known as a secret key -is shared only with key's initiator. When someone wants to send an encrypted message, they can pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can then decrypt the message using their related private key. If the sender encrypts the message using their private key, the message can be decrypted only using that sender's public key, thus authenticating the sender. These encryption and decryption processes happen automatically; users do not need to physically lock and unlock the message.

Many protocols rely on asymmetric cryptography, including the transport layer security (TLS) and secure sockets layer (SSL) protocols, which make HTTPS possible. The encryption process is also used in software programs that need to establish a secure connection over an insecure network, such as browsers over the internet, or that need to validate a digital signature.

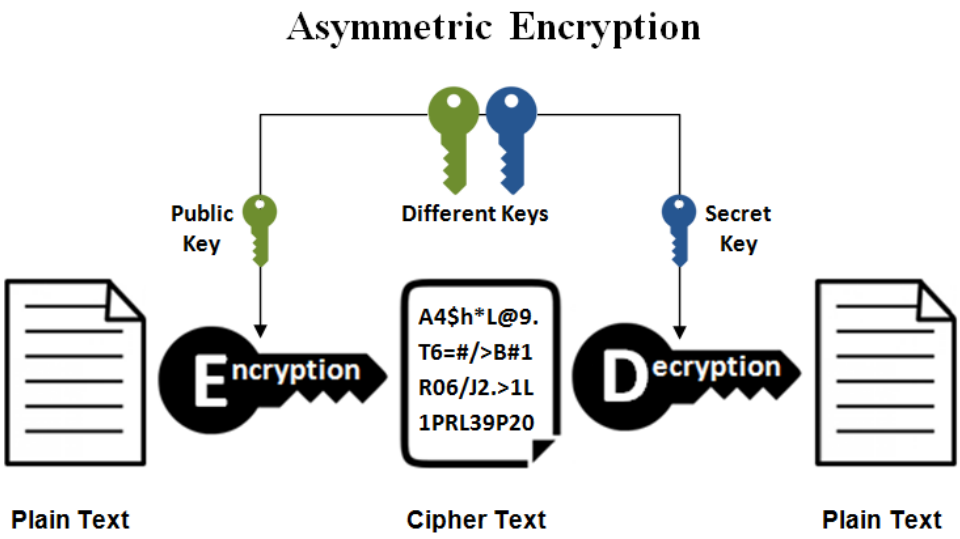

Figure 1.3 : Asymmetric Encryption [3]

Chapter 2

Data Encryption Standard

2.1 History of DES

The main standard for encrypting data was a symmetric algorithm known as the Data Encryption Standard (DES). However, this has now been replaced by a new standard known as the Advanced Encryption Standard (AES). DES is a 64 bit block cipher which means that it encrypts data 64 bits at a time. This is contrasted to a stream cipher in which only one bit at a time (or sometimes small groups of bits such as a byte) is encrypted.

DES was the result of a research project set up by International Business Machines (IBM) corporation in the late 1960's which resulted in a cipher known as LUCIFER. In the early 1970's it was decided to commercialise LUCIFER and a number of significant changes were introduced. IBM was not the only one involved in these changes as they sought technical advice from the National Security Agency (NSA) (other outside consultants were involved but it is likely that the NSA were the major contributors from a technical point of view). The altered version of LUCIFER was put forward as a proposal for the new national encryption standard requested by the National Bureau of Standards (NBS). It was finally adopted in 1977 as the Data Encryption Standard - DES (FIPS PUB 46).

Some of the changes made to LUCIFER have been the subject of much controversy even to the present day. The most notable of these was the key size. LUCIFER used a key size of 128 bits however this was reduced to 56 bits for DES. Even though DES actually accepts a 64 bit key as input, the remaining eight bits are used for parity checking and have no effect on DES's security. Outsiders were convinced that the 56 bit key was an

easy target for a brute force attack due to its extremely small size. The need for the parity checking scheme was also questioned without satisfying answers

Another controversial issue was that the S-boxes used were designed under classified conditions and no reasons for their particular design were ever given. This led people to assume that the NSA had introduced a “trapdoor” through which they could decrypt any data encrypted by DES even without knowledge of the key. One startling discovery was that the S-boxes appeared to be secure against an attack known as Differential Cryptanalysis which was only publicly discovered by Biham and Shamir in 1990. This suggests that the NSA were aware of this attack in 1977; 13 years earlier! In fact the DES designers claimed that the reason they never made the design specifications for the S-boxes available was that they knew about a number of attacks that weren’t public knowledge at the time and they didn’t want them leaking - this is quite a plausible claim as differential cryptanalysis has shown. However, despite all this controversy, in 1994 NIST reformed DES for government use for a further five years for use in areas other than “classified”.

DES of course isn’t the only symmetric cipher. There are many others, each with varying levels of complexity. Such ciphers include: IDEA, RC4, RC5, RC6 and the new Advanced Encryption Standard (AES). AES is an important algorithm and was originally meant to replace DES (and its more secure variant triple DES) as the standard algorithm for non-classified material. However as of 2003, AES with key sizes of 192 and 256 bits has been found to be secure enough to protect information up to top secret. Since its creation, AES had underdone intense scrutiny as one would expect for an algorithm that is to be used as the standard. To date it has withstood all attacks but the search is still on and it remains to be seen whether or not this will last.

2.2 DES Encryption

The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds. Each round uses a different 48-bit round key generated from the cipher key according to a predefined algorithm.

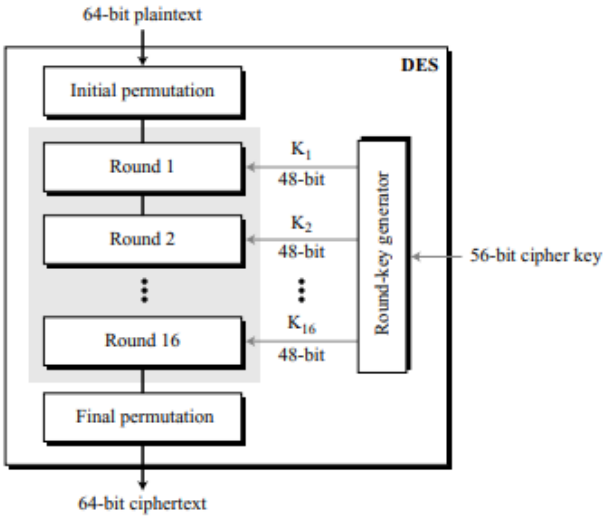

Figure 2.1: General Structure of DES [8]

Initial and Final Permutations

Figure 2.2 shows the initial and final permutations (P-boxes). Each of these permutations takes a 64-bit input and permutes them according to a predefined rule. We have shown only a few input ports and the corresponding output ports. These permutations are keyless straight permutations that are the inverse of each other. For example, in the initial permutation, the 58th bit in the input becomes the first bit in the output. Similarly, in the final permutation, the first bit in the input becomes the 58th bit in the output. In other words, if the rounds between these two permutations do not exist, the 58th bit entering the initial permutation is the same as the 58th bit leaving the final permutation.

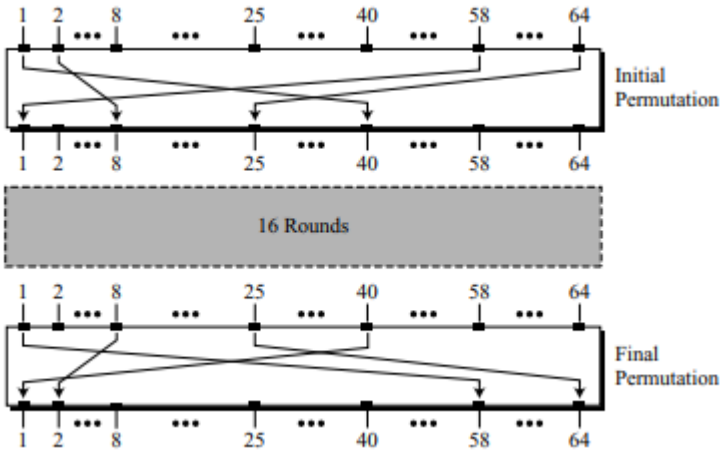

Figure 2.2: Initial and Final Permutations [8]

The permutation rules for these P-boxes are shown in Table 2.1. Each side of the table can be thought of as a 64-element array. Note that, as with any permutation table we have discussed so far, the value of each element defines the input port number, and the order (index) of the element defines the output port number.

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Table 2.1: Initial and Final Permutation Table

Rounds

DES uses 16 rounds. Each round of DES is a Feistel cipher, as shown in Figure 2.3

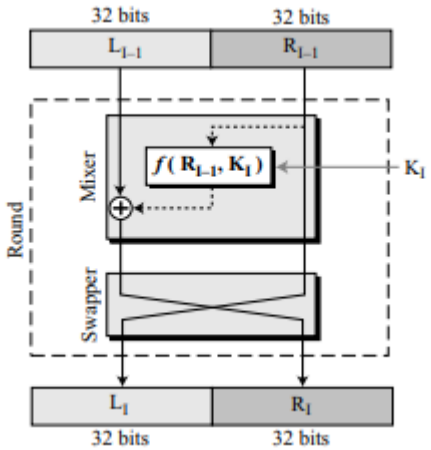

Figure 2.3: Rounds of DES [8]

The round takes LI−1 and RI−1 from the previous round (or the initial permutation box) and creates LI and RI, which go to the next round (or final permutation box). We can assume that each round has two cipher elements (mixer and swapper). Each of these elements is invertible. The swapper is invertible. It swaps the left half of the text with the right half. The mixer is invertible because of the XOR operation. All noninvertible elements are collected inside the function f (RI−1, KI).

DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits (RI−1) to produce a 32-bit output. This function is made up of four-sections: an expansion P-box, a whitener (that adds key), a group of S-boxes, and a straight P-box as shown in Figure 2.4.

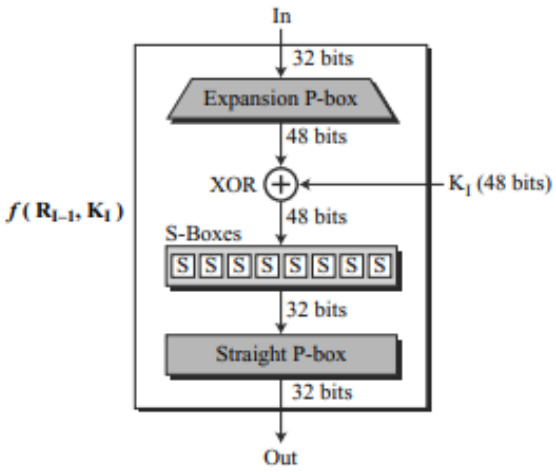

Figure 2.4: DES Function [8]

Expansion P-Box

Since RI−1 is a 32-bit input and KI is a 48-bit key, we first need to expand RI−1 to 48 bits. RI−1 is divided into 8 4-bit sections. Each 4-bit section is then expanded to 6 bits. This expansion permutation follows a predetermined rule. For each section, input bits 1, 2, 3, and 4 are copied to output bits 2, 3, 4, and 5, respectively. Output bit 1 comes from

bit 4 of the previous section; output bit 6 comes from bit 1 of the next section. If sections 1 and 8 can be considered adjacent sections, the same rule applies to bits 1 and 32. Figure 2.5 shows the input and output in the expansion permutation.

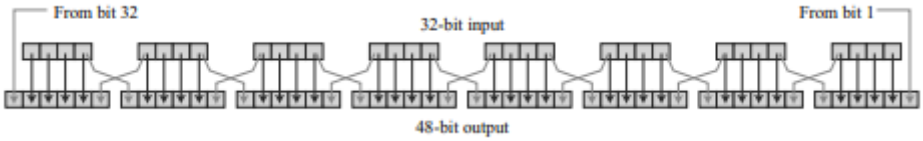

Figure 2.5: Expansion P-Box [8]

Although the relationship between the input and output can be defined mathematically, DES uses Table 2.2 to define this P-box. Note that the number of output ports is 48, but the value range is only 1 to 32. Some of the inputs go to more than one output. For example, the value of input bit 5 becomes the value of output bits 6 and 8.

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

Table 2.2: Expansion P-Box Table

Whitener (XOR)

After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key. Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.

S – Boxes

The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 2.6.

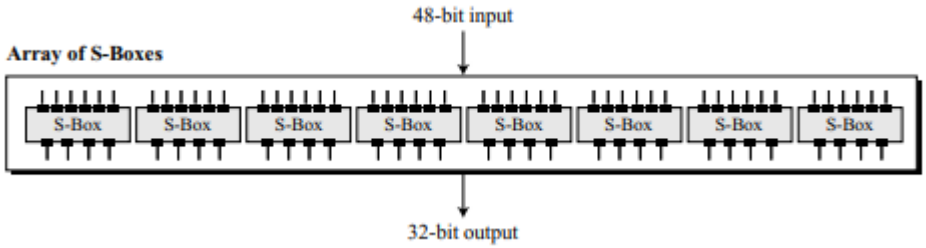

Figure 2.6: S-Boxes [8]

The 48-bit data from the second operation is divided into eight 6-bit chunks, and each chunk is fed into a box. The result of each box is a 4-bit chunk; when these are combined the result is a 32-bit text. The substitution in each box follows a pre-determined rule based on a 4-row by 16-column table. The combination of bits 1 and 6 of the input defines one of four rows; the combination of bits 2 through 5 defines one of the sixteen columns as shown in Figure 2.7. Because each S-box has its own table, we need eight tables, to define the output of these boxes. The values of the inputs (row number and column number) and the values of the outputs are given as decimal numbers to save space. These need to be changed to binary.

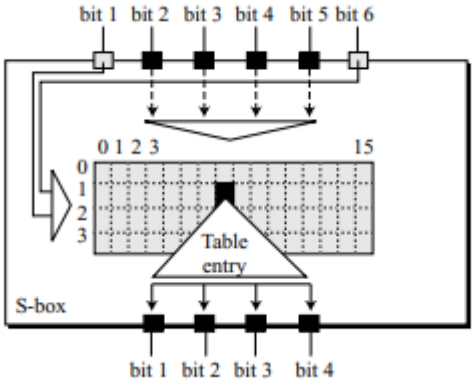

Figure 2.7: S-Box Rule [8]

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Table 2.3: S-Box 1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

Table 2.4: S-Box 2

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

Table 2.5: S-Box 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	07	13	14	03	00	6	09	10	1	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

Table 2.6: S-Box 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

Table 2.7: S-Box 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	10	00	08	13

Table 2.8: S-Box 6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

Table 2.9: S-Box 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	10	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	10	15	03	05	08
3	02	01	14	07	04	10	8	13	15	12	09	09	03	05	06	11

Table 2.10: S-Box 8

Straight Permutation

The last operation in the DES function is a straight permutation with a 32-bit input and a 32-bit output. The input/output relationship for this operation is shown in Table 2.3 and follows the same general rule as previous permutation tables. For example, the seventh bit of the input becomes the second bit of the output.

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Table 2.11: Straight Permutation Table

Cipher and Reverse Cipher

Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds. The cipher is used at the encryption site; the reverse cipher is used at the decryption site. The whole idea is to make the cipher and the reverse cipher algorithms similar.

To achieve this goal, one approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper. This is done in Figure 2.8.

Although the rounds are not aligned, the elements (mixer or swapper) are aligned. The final and initial permutations are also inverses of each other. The left section of the plaintext at the encryption site, L0, is enciphered as L16 at the encryption site; L16 at the decryption is deciphered as L0 at the decryption site. The situation is the same with R0 and R16.

A very important point we need to remember about the ciphers is that the round keys (K1 to K16) should be applied in the reverse order. At the encryption site, round 1 uses K1 and round 16 uses K16; at the decryption site, round 1 uses K16 and round 16 uses K1.

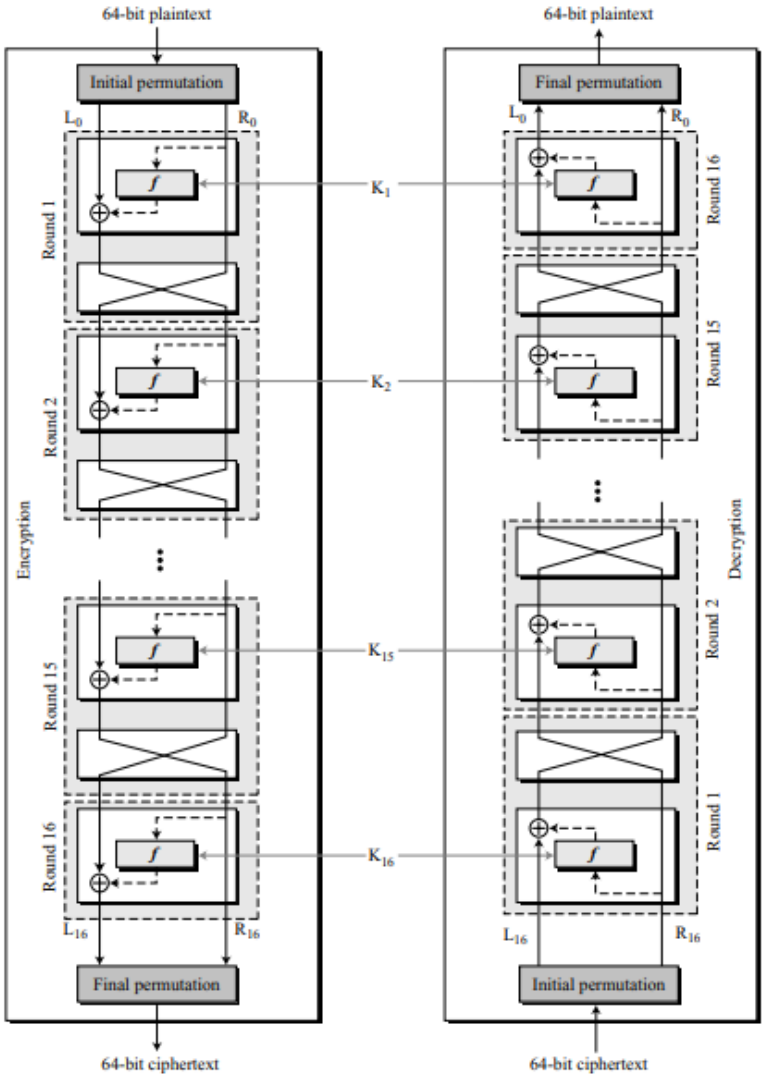

Figure 2.8: Cipher and Reverse Cipher [8]

DES Key generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. However, the cipher key is normally given as a 64-bit key in which 8 extra bits are the parity bits, which are dropped before the actual key-generation process, as shown in Figure 2.9

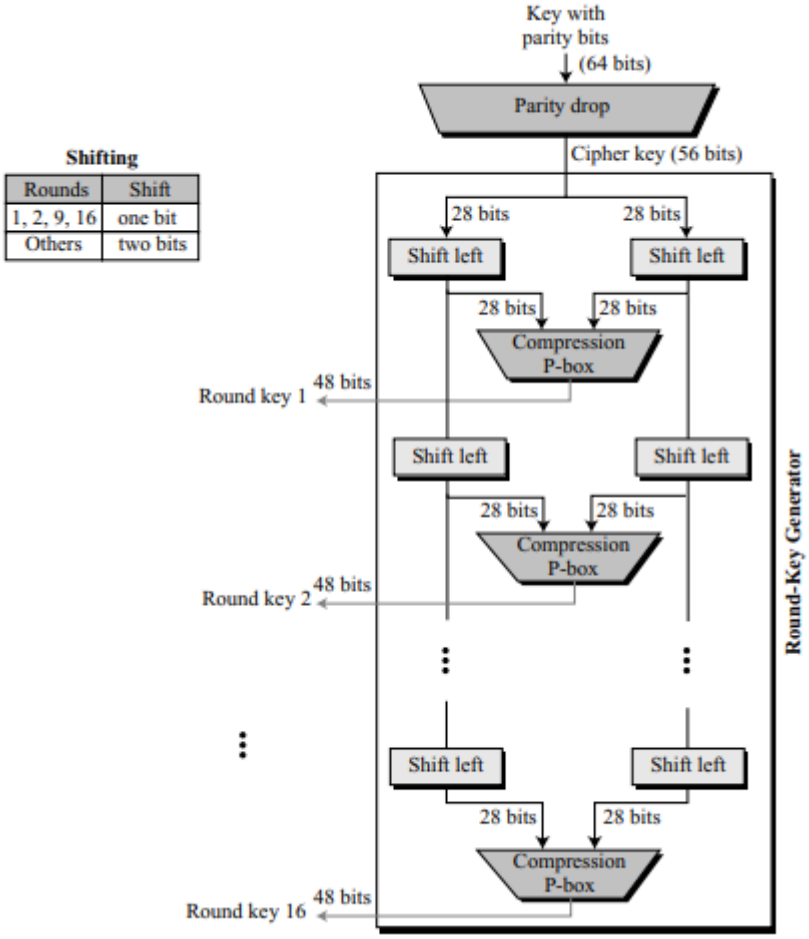

Figure 2.9: Key Generation [8]

Parity Drop

The preprocess before key expansion is a compression permutation that we call parity bit drop. It drops the parity bits (bits 8, 16, 24, 32, …, 64) from the 64-bit key and permutes the rest of the bits according to Table 6.12. The remaining 56-bit value is the actual cipher key which is used to generate round keys. The parity drop permutation (a compression P-box) is shown in Table 6.12.

Shift Left

After the straight permutation, the key is divided into two 28-bit parts. Each part is shifted left (circular shift) one or two bits. In rounds 1, 2, 9, and 16, shifting is one bit; in the other rounds, it is two bits. The two parts are then combined to form a 56-bit part. Table 6.13 shows the number of shifts for each round.

Compression Permutation

The compression permutation (P-box) changes the 58 bits to 48 bits, which are used as a key for a round. The compression permutation is shown in Table 6.14.

Avalanche Effect

The avalanche effect is a term associated with a specific behaviour of mathematical functions used for encryption. Avalanche effect is considered as one of the desirable property of any encryption algorithm. A slight change in either the key or the plain-text should result in a significant change in the cipher-text. This property is termed as avalanche effect.

In simple words, it quantifies the effect on the cipher-text with respect to the small change made in plain text or the key.

In case of algorithm that uses hash value, even a small alteration in an input string should drastically change the hash value. In other words, flipping single bit in input string should at least flip half of the bits in the hash value.

A good encryption algorithm should always satisfy the following relation:

Avalanche effect > 50%

The effect ensures that an attacker cannot easily predict a plain-text through a statistical analysis. An encryption algorithm that doesn't satisfies this property can favour an easy statistical analysis. That is, if the alteration in a single bit of the input results in change of only single bit of the desired output, then it's easy to crack the encrypted text.

If a block cipher or cryptographic hash function does not exhibit the avalanche effect to a significant degree, then it has poor randomization, and thus a cryptanalyst can make

predictions about the input, being given only the output. This may be sufficient to partially or completely break the algorithm. Thus, the avalanche effect is a desirable condition from the point of view of the designer of the cryptographic algorithm or device.

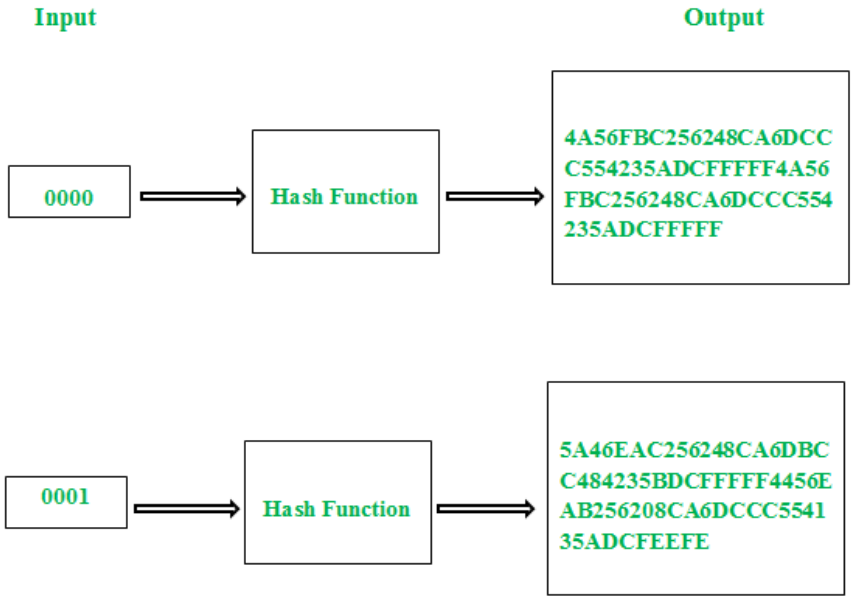

Figure 2.10: Avalanche Effect [4]

Chapter 3

Proposed Work

3.1 Technical Details

1. **Python**

   Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). It is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

2. **SageMath Cell**

   The software used in this project is SageMath Cell. It is an open source implementation of mathematics and scientific software based on Python. It contains many existing open-source packages: Numpy, SciPy, matplotlib, etc. The documentation of cryptography in sagemath is very resourceful for the project. Prerequisites for SageMath installation:

   1. A 64-bit version of Windows with at least 6 GB of free disk space and at least 2 GB of RAM is needed.
   2. The following standard command-line development tools must be installed on the system: C/C++ compiler version 5.1 or later, GNU make

version 3.82 or later, GNU m4 1.4.2 or later, Perl version 5.8.0, GNU tar version 1.17 or later, Python version 3.7 or later

3. Jupyter notebook has to be pre installed to run Sage Math notebook.

DES algorithm used 64-bit keys, after dropping the parity bits only 56 bits of the key is used to encrypt the data. For a modern-day computer, a brute force attack using 2^{56} keys is an easy nut to crack. In order to try to overcome this problem, DES key generation algorithm is modified in this project to use 128-bit keys.

3.2 Methodology

1. Firstly, DES code was written.
2. Next, modified DES key generation function was coded.
3. Then standard DES and modified DES were analysed by comparing avalanche effect on them.

3.2.1 DES Encryption Algorithm

1. In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
2. The initial permutation is performed on plain text.
3. Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
4. Now each LPT and RPT go through 16 rounds of the encryption process.
5. In the end, LPT and RPT are re-joined and a Final Permutation (FP) is performed on the combined block
6. The result of this process produces 64-bit ciphertext.

3.2.2 DES Key Generation Algorithm

1. Key of size 64-bit is input to key generation function.
2. The key goes through a compression permutation which is called parity bit drop. It drops the parity bits (bits 8, 16, 24, 32, …, 64) from the 64-bit key and permutes the rest of the bits according to Table 1.
3. The remaining 56-bit key is split into two 28-bit parts.
4. An empty array is created to store round keys.
5. Left part and right part of key are shifted left (circular shift) one or two bits. In rounds 1, 2, 9, and 16, shifting is one bit; in the other rounds, it is two bits. Table 6.13 shows the number of shifts for each round.
6. The 28-bit keys are combined to get the actual 56-bit cipher key which is used to generate round keys.
7. The 56 bits key is compressed to get 48 bits key, using the compression permutation (P-box) which is used as a key for a round. The P-box is shown in Table. The key generated is first round key and it is added to round key array.
8. Steps 6-8 are repeated 15 more times and every time the key generated is added to round key array.
9. The round key array is output of this function.

3.2.3 Avalanche effect Algorithm

1. Two cipher text which are compared to get avalanche effect are input to the function.
2. Each element of both cipher text is compared, if they are found same a variable count is incremented by one.
3. In order to get number of different characters, count variable was subtracted from length of cipher text.
4. Avalanche effect is calculated according to following equation,
   Avalanche effect = number of different character * 100/ length of cipher text

3.2.4 Modified Key Generation Algorithm:

1. Key of size 128-bit is input to key generation function.
2. The first step is to divide the key into two 64-bit parts.
3. Both parts of key go through a compression permutation which is called parity bit drop. It drops the parity bits (bits 8, 16, 24, 32, …, 64) from the 64-bit key and permutes the rest of the bits according to Table 1.
4. From each of the two remaining 56-bit keys, 28-bits are selected using random function of python.
5. An empty array is created to store round keys.
6. Left part and right part of key are shifted left (circular shift) one or two bits. In rounds 1, 2, 9, and 16, shifting is one bit; in the other rounds, it is two bits. Table 6.13 shows the number of shifts for each round.
7. The 28-bit keys are combined to get the actual 56-bit cipher key which is used to generate round keys.
8. The 58 bits key is compressed to get 48 bits key, using the compression permutation (P-box) which is used as a key for a round. The P-box is shown in Table 6.14. The key generated is first round key and it is added to round key array.
9. Steps 6-8 are repeated 15 more times and every time the key generated is added to round key array.
10. The round key array is returned.

Chapter 4

Results

4.1 Avalanche effect on changing first character

1. With Hex as input on des and modified des algo

    a) Input array of 64-bit Hexadecimal text with only first character different was made. First character ranges from A-F and 0-9.
    b) 64-bit key is given as input to DES key generation function and round key array is the output.
    c) DES encryption function is called with element of hexadecimal array and round key array as inputs. Same round key array is used for all inputs to calculate avalanche effect.
    d) Cipher text of each input element in both binary and hexadecimal form, is stored in an array.
    e) Avalanche effect for all cipher text is calculated by comparing it with cipher text of first input then average of all avalanche effect is calculated.
    f) Above steps are repeated for modified DES algorithm with 128-bit key as input for modified key generation function.

Cipher text form	DES	Modified DES
Hexadecimal	93.33	95.83
Binary	50.729	51.15

Table 4.1 Avalanche effect on changing last character with hexadecimal input

2. With Plain text as input on des and modified des.

a) Input array of 64-bit plain text with only first character different was made. First character ranges from A-Z, a-z and 0-9.
b) 64-bit key is given as input to DES key generation function and round key array is the output.
c) DES encryption function is called with element of input array and round key array as inputs. Same round key array is used for all inputs to calculate avalanche effect.
d) Cipher text of each input element in both binary and hexadecimal form, is stored in an array.
e) Avalanche effect for cipher text of every input (except first) is calculated by comparing it with cipher text of first input then average of all avalanche effect is calculated.
f) Above steps are repeated for modified DES algorithm with 128-bit key as input for modified key generation function.

Cipher text form	**DES**	**Modified DES**
Hexadecimal	92.62	93.03
Binary	49.51	49.95

Table 4.2 Avalanche effect on changing First character with plain text input

4.2 Avalanche effect on changing last character

Above two algorithms were repeated for changing last character of input.

1. With Hex as input on des algo and modified des algo

    a) Input array of 64-bit Hexadecimal text with only last character different was made. First character ranges from A-F and 0-9.
    b) 64-bit key is given as input to DES key generation function and round key array is the output.
    c) DES encryption function is called with element of hexadecimal array and round key array as inputs. Same round key array is used for all inputs to calculate avalanche effect.
    d) Cipher text of each input element in both binary and hexadecimal form, is stored in an array.
    e) Avalanche effect for all cipher text is calculated by comparing it with cipher text of first input then average of all avalanche effect is calculated.
    f) Above steps are repeated for modified DES algorithm with 128-bit key as input for modified key generation function.

Cipher text form	DES	Modified DES
Hexadecimal	91.66	94.16
Binary	50.62	47.91

Table 4.3 Avalanche effect on changing last character with hexadecimal input

2. With Plain text as input on des algo and modified des algo

    a) Input array of 64-bit plain text with only last character different was made. First character ranges from A-Z, a-z and 0-9.

b) 64-bit key is given as input to DES key generation function and round key array is the output.
c) DES encryption function is called with element of input array and round key array as inputs. Same round key array is used for all inputs to calculate avalanche effect.
d) Cipher text of each input element in both binary and hexadecimal form, is stored in an array.
e) Avalanche effect for cipher text of every input (except first) is calculated by comparing it with cipher text of first input then average of all avalanche effect is calculated.
f) Above steps are repeated for modified DES algorithm with 128-bit key as input for modified key generation function.

Cipher text form	**DES**	**Modified DES**
Hexadecimal	95.28	94.47
Binary	50.614	50.15

Table 4.4 Avalanche effect on changing last character with plain text input.

Chapter 5

Conclusion & Future Work

Critics believe that the most serious weakness of DES is in its small key size. To do a brute-force attack on a given ciphertext block, the adversary needs to check 256 keys. With available technology it is possible in few days, but with quantum computers which are on the rise it will be a matter of minutes. This study shows increasing the key size to 128 bits leads to increase in avalanche effect which is a desirable property for a cryptography algorithm.

In this project DES was modified by changing key generation algorithm. On analysis, the results show that modified DES has higher avalanche effect compared to DES when first character changes in input text. But modified DES has lower avalanche effect compared to DES when last character changes in input text.

We intend to extend this study, by increasing the key size up to 256 bit and performing cryptanalysis on the proposed algorithm to improve it. Due to limitation of time and resources the study couldn’t be progressed further.

This page was intentionally left blank.

References

[1] Sagemath cryptography documentation- https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/block_cipher/des.html

[2] Cryptography - https://www.synopsys.com/glossary/what-is-cryptography.html/

[3] Cryptography Figure - https://www.techtarget.com/searchsecurity/definition/cryptography

[4] Symmetric and Asymmetric Cryptography Figure - https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences

[5] Avalanche Effect Figure - https://www.geeksforgeeks.org/avalanche-effect-in-cryptography/

[6] Avalanche Effect - https://en.wikipedia.org/wiki/Avalanche_effect#:~:text=In%20cryptography%2C%20the%20avalanche%20effect,half%20the%20output%20bits%20flip).

[7] History of DES - http://www.umsl.edu/~siegelj/information_theory/projects/des.netau.net/des%20history.html

[8] Behrouz A. Forzan, “INTRODUCTION TO CRYPTOGRAPHY AND NETWORK SECURITY”, First Edition, McGrawHill Higher Education, New Delhi, 2007

[9] Python - https://www.tutorialspoint.com/python/index.html

[10] Bhargavi Goswami, “Study and Analysis of Symmetric Key-Cryptograph DES, Data Encryption Standard”, Proceedings of UGC sponsored National Seminar on Scientific Wealth of Physics SWP-2012, August 2012