QATAR UNIVERSITY
COLLEGE OF ENGINERING
CSE DEPARTMENT

CMPE 485 Computer Security L01
CRN - 29955
Dr. Ryan Riley
SPRING 2016

Term Project
DES Encryption

STUDENT NAME: Babikr Elnimah
STUDENT ID: 201204005

STUDENT NAME: Mohammed Amaan Sheikh
STUDENT ID: 201205380

STUDENT NAME: Ahmed Ibrahim
STUDENT ID: 201204361

Table of Contents

Abstract……..3

Introduction……3-4

DES Algorithm Definition……4-10

Weaknesses of DES……10-11

How Newer Standards Addressed the Weaknesses….11

Conclusion……12

References……13

Abstract

Encryption can be defined as the procedure of altering data in a manner that makes it unusable to any entity except those who possess the key required to restore the information to its original form. As a result, encryption is vital security measure used to ensure the safe passage of data from point to point. The purpose of exercising this protective measure is to keep data of all varieties safe from prying eyes. As such, many means have been developed in order to achieve this essential task. Consequently, businesses, governments, and individuals use encryption to protect various sensitive data such as corporate secrets, classified information, and personal information. However, as the needs of those who use encryption differ so do the encryption methods in order to accommodate the various manners in which they are deployed.

Consequently, the next logical step in solving this problem is selecting an encryption algorithm in which we can encrypt the data, for the purposes of this report we chose DES (Data Encryption Standard) due to the fact that until recently it was the de-facto standard for encrypting data. Through the course of this report we did an in detail analysis of what DES is and explained how it functions as an algorithm. Moreover, we explored the history of DES and how it came to fruition as well as who created and why. In addition, we investigated the weaknesses of DES and how it was broken. Lastly, we looked into how future encryption algorithms fixed the weak points of DES.

Introduction

DES (Data Encryption Algorithm) has an interesting history in how it came to be. According to umsl.edu in the late 1960's IBM commenced a research project that resulted in the creation of an encryption cipher known as Lucifer. Approximately a decade later IBM

made the decision to monetize their cipher. Therefore, in order achieve this they enlisted the assistance of several third party entities. However, in terms of technical expertise the main contributing body was the NSA (National Security Agency). Around the same time period at which this collaborative effort was ongoing the NBS (National Board of Standards) was seeking a new standard for the purposes of data encryption. Consequently, IBM submitted the altered version of Lucifer in hopes of answering the NBS's call. Finally, in 1977 the NBS declared the altered version of Lucifer DES and adopted it as its namesake.

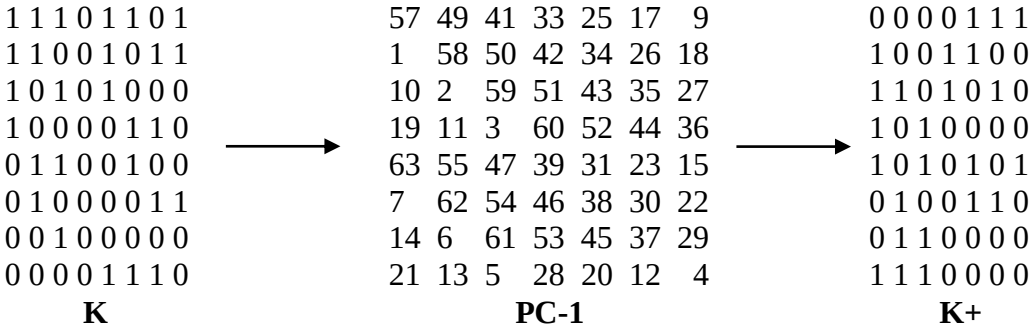
As time moved along so did the progression of DES encryption. The most recent development in which is the advent of 3DES or triple DES. 3DES is processes of running DES three times sequentially with a new key in each round for the purpose of increasing the bit entropy of the encryption.

DES Algorithm Definition

DES is an encryption algorithm that uses a key size of 64-bits and encrypts a 64-bit block of plain text into a corresponding 64-bit block of cipher text. To do this, it first needs to convert the plain text into binary bits by using ASCII mapping (American Standard Code for Information Interchange). ASCII is a character encoding standard that allows you to represent any character as an 8-bit binary number. Thus, 8 characters are required to form a 64-bit binary value.

The first step in the DES algorithm is to generate 15 sub-keys from the original key. This is done by going through a series of permutations. A permutation is defined as an act of rearranging the elements of a set according to some sequence or order. In our context, it means to rearrange the 64 bits of the key according to a given sequence. The first permutation that is performed on the key is called as the PC-1 (Permuted Choice 1). This part of the algorithm takes the 64-bit key as input and gives an output of 56-bits. This is because every 8^{th} bit of the key is ignored in the encryption algorithm. However, 8^{th} bits that are ignored are

used as parity bits (meant for error detection). The PC-1 operates as follows :



The above illustration shows that the original 64-bit key (K) goes through the permutation block (PC-1) that outputs another 56-bit key (K+). The first bit of K+ is the 57^{th} bit of K, the second bit is the 49^{th} and so on until the last bit is the 4^{th} bit of K.

After deriving K+ from the original key, it is then further divided into two halves called C_0 and D_0, each consisting of 28 bits respectively. From this pair, we then derive 16 more pairs by doing a series of bit rotations. For each iteration, the number of bit rotations are as follows:

Iteration Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

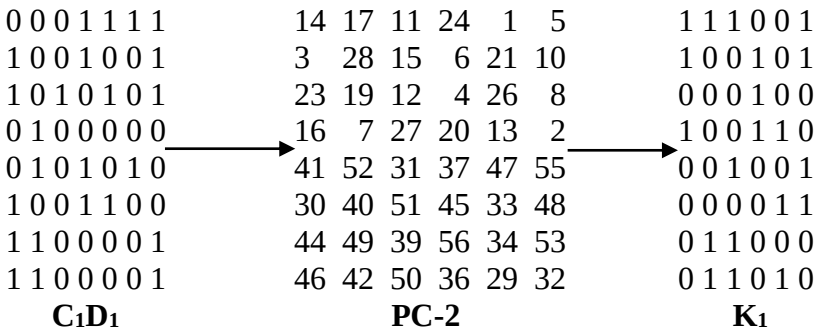
Every C_nD_n pair is obtained by performing the corresponding number of left rotations on the previous pair $C_{n-1}D_{n-1}$. This leaves us with 16 different pairs of C_nD_n.

C_1 = 000111100110011010101010100000 D_1 = 010101010011001100000111100001

. .
. .
. .

C_{16} = 000011110011001101010101010000 D_{16} = 101010101001100110000111000 0

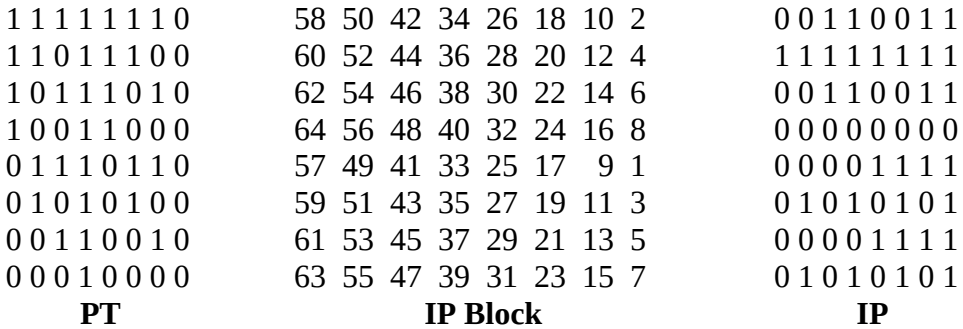
Each of these is then permuted using another permutation table called PC-2 (Permuted Choice 2). This permutation table takes as an input each of the 56 bit C_nD_n keys and outputs a set of 48 bit (K_n) keys corresponding to each key inputted.



As can be seen from the illustration above, the first bit of K_n is the 14^{th} bit of C_nD_n and so on. At this point in the encryption algorithm, we have 16 keys of 48 bits each. Now we can start encrypting each 64-bit block of plain text.

The DES algorithm uses a Fiestal network which divides the plain text into a right half and the left half. It then performs an XOR operation on the left half and then switches the right half to the left and vice-versa. This is known as one Fiestal round. The DES encryption algorithm performs 16 Fiestal rounds. We intend to show how this is accomplished in the next few steps.

We start by taking our 64-bit block of plain text and running it through another permutation called IP (Initial Permutation). The IP block of the algorithm takes as input 64-bits of plain text(PT) and outputs 64-bits of IP. This is illustrated below.



As in every other permutation, the first bit of the IP is the 58^{th} bit of the PT and so on. This IP is then divided into 2 halves which are labeled L_0 and R_0 respectively. These two halves of 32 bits each are then fed into the first round of the Fiestal network. The Fiestal network performs an operation on the left half of the IP and feeds it to the right side input of

the next round. Meanwhile the right half is sent to the left half of the IP without any changes.

$$L_n = R_{n-1} \qquad \& \qquad R_n = L_{n-1} + f(R_{n-1}, K_n)$$

This goes on for 16 rounds of the Fiestal cycle. For each round of the cycle, the corresponding key is used. The left half of the function is XORed with a function (**f**) which takes as input the right half and the corresponding Key. What this function does is extremely complex. Let us look at it closely.

The function f takes the right side of the message (32 bits) and runs it through an E bit-selection table to permute it further. The E bit-selection table take 32-bits as input and outputs a 48-bit block that repeats some of the bits. This is illustrated below.

R_{n-1}	**E bit-selection**	$E(R_{n-1})$
0 0 0 0	32 1 2 3 4 5	1 0 0 0 0 1
1 1 1 1	4 5 6 7 8 9	0 1 1 1 1 0
0 1 0 1	8 9 10 11 12 13	1 0 1 0 1 0
0 1 0 1	12 13 14 15 16 17	1 0 1 0 1 0
0 0 0 0	16 17 18 19 20 21	1 0 0 0 0 1
1 1 1 1	20 21 22 23 24 25	0 1 1 1 1 0
0 1 0 1	24 25 26 27 28 29	1 0 1 0 1 0
0 1 0 1	28 29 30 31 32 1	1 0 1 0 1 0

The output of this block is the 48 bit $E(R_{n-1})$ which is then XORed with the corresponding 48-bit key K_n. The resultant 48 bits are then divided into 8 different groups of six bits each.

$$E(R_{n-1}) \text{ XOR } K_n = B_1B_2B_3B_4B_5B_6B_7B_8$$

Each of these group of 6 bits is run through a different S box. There are 8 S boxes in total. Each of them take one of the 6-bit B as input and give an output of 4 bits. The 8 S boxes are shown below.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

Each of the S_n boxes takes the 6 bit B_n block as input and uses the first and last bit of the number to represent a 2-bit number that ranges from 0 to 3. This defines the row of the S_n box. The remaining 4 bits represent a 4-bit number that ranges from 0 to 15. This defines the

column of the S_n box. The output is the corresponding element of the specified row and column of the S_n box concerned. This output is a number between 0 and 15 and thus translates to a 4-bit binary value. After finding the output from each of the 8 S boxes, these values are combined to give a 32-bit number. This 32-bit value is then run through another permutation table called the P function. This function takes 32-bits as input and gives a 32-bit output. This is illustrated below.

1 0 1 0	16 7 20 21	1 1 0 1
0 0 1 1	29 12 28 17	1 1 0 0
0 1 1 1	1 15 23 26	1 0 1 1
1 1 0 1	5 18 31 10	0 1 0 1
0 1 0 0	2 8 24 14	0 1 0 1
1 0 1 0	32 27 3 9	0 1 1 0
0 1 1 0	19 13 30 6	0 1 0 0
1 0 0 0	22 11 4 25	0 1 0 0
S	**P table**	**f**

This 32-bit output is also the output of the function f. So far, we have illustrated how to calculate the value of the function f and how it is used in the Fiestel network. The output of this Fiestal network, $R_{16}L_{16}$ is run through a final permutation that is called the IP^{-1}. This is actually the inverse of the initial permutation that was done previously. It takes 64 bits as input and outputs a 64-bit cipher text. This is illustrated below.

1 1 1 1 0 1 0 1	40 8 48 16 56 24 64 32	0 1 1 1 1 0 1 0
1 0 1 1 0 0 1 1	39 7 47 15 55 23 63 31	0 0 0 1 0 1 1 1
0 0 1 0 0 1 1 0	38 6 46 14 54 22 62 30	1 1 1 0 1 1 0 0
0 1 1 0 1 0 1 0	37 5 45 13 53 21 61 29	1 0 1 0 1 0 1 1
1 0 1 1 1 1 0 0	36 4 44 12 52 20 60 28	1 1 1 1 0 0 0 0
1 0 1 1 1 1 0 1	35 3 43 11 51 19 59 27	1 1 1 1 0 1 0 1
1 1 0 0 1 1 0 1	34 2 42 10 50 18 58 26	0 1 0 0 1 0 1 1
1 1 0 0 1 0 1 1	33 1 41 9 49 17 57 25	1 1 1 1 1 0 1 0
$R_{16}L_{16}$	**IP^{-1}**	**CT**

This is done for each block of 64-bit plain text.

Decryption works in the exactly same way except that it uses the keys K_1 to K_{16} in the reverse order. This means that for the first round of the Fiestal network we will use K_{16}, for

the second K_{15} and so on until the last round of the Fiestal network which will use K_1. This will give you back the plain text.

Weaknesses of DES

Over time as computers became faster and faster, they became more accessible and the weaknesses of DES began to become problematic. Some will argue that these weaknesses were problematic from the advent of the algorithm however there is little argument nowadays that DES is secure. To understand what the most significant issue with DES is one has to accept the fact that encryption strength is decidedly linked to key sizes. This is due to the fact that, as the size of the key increases so does the difficulty of breaking said key. The property that allows for this is that with every increase, of a single bit, in key size the pool of potential keys increases exponentially. For instance, if an algorithm has a 10-bit key it will have a potential pool of 1024 keys (2^{10}). Meanwhile, an 11-bit key algorithm will have a potential pool of 2048 keys (2^{11}). DES's critical flaw is that despite having a 64-bit key, which is also considered weak by today's standards, the fact that every 8^{th} bit is a checksum essentially reduces the effective key size to 56-bits. Thus, an attacker would have to attempt a maximum of 7.206×10^{16} (2^{56}) keys in order guess the key (worst case scenario) or 3.603×10^{16} to have 50% percent chance of doing so (Average Case). To put this is perspective, "Chips to perform one million of DES encrypt or decrypt operations a second are available (in 1993). A $1 million DES cracking machine can search the entire key space in about 7 hours (Kapoor, Mohan, & Kumar, 2012)."

This begs two questions, the first of which being why would someone spend 1 million dollars to break DES? The answer lies in the answer of another question - Who uses DES? The US Government still uses DES to encrypt data until this day. Granted that although they have mostly switched over to using 3DES, not the entire government has. Thus, individuals and organizations with the capital to build such a machine would be incentivized to do so as

the US Government is a target rich environment. The second question being that, if a computer can try 1 million encrypts and decrypts per second it would still take it 1142 years to break DES. Therefore, wouldn't it still be secure? One has to keep in mind that 1 million encrypts and decrypts per second were the abilities of a chip in 1993. Due to Moore's Law of the doubling of transistors on microchips this value has sharply increased in past 13 years. Modern computers can do in excess of 292 billion calculations per second. Therefore, DES has become vulnerable enough to attacks over the years to warrant becoming obsolete. Lastly, another point of concern for DES is the fact that it uses symmetric crypto meaning that the decryption key and encryption key are the same. Thus, it is susceptible to other attacks that take advantage of this fact.

How Newer Standards Addressed the Weaknesses

Having established that DES was badly broken namely due to its inferior key size a newer standard was required in order to succeed DES. AES (Advanced Encryption Standard) is the encryption algorithm that took that honor. The main means in which AES solved the deficiencies of DES was by increasing the key size to a minimum of 128-bits. It also gave users the option to use 192 or 256 bit keys for the extra paranoid. As result, brute force attacks we rendered unfeasible. Thus, in effect neutralizing the largest threat to DES. Moreover, AES added the ability to encrypt 128-bit (16-byte) blocks of data as compared to DES's 64-bits. Lastly, the Feistel Network that is used by DES in order encrypt data introduced a vulnerability as it was did not designed to be used with algorithms that are incapable of producing pseudo random keys. Therefore, due to the fact that all the sub keys that are used in the encryption process of DES were based on the main key this posed a significant threat. AES also mitigated this by instead using a similar iterative loop based permutation system however it does not use the Feistal network based approach.

Conclusion

In conclusion, DES is interesting case study of the significance of encryption. This is evident when piecing together the findings of this report. For instance, historically speaking at time it was developed it met the standards of what was required by an encryption algorithm. However, due to a lack of foresight DES became a ticking time bomb that inevitably blew up. For all its faults though DES left a lasting legacy as it is regarded as being the predecessor to AES the modern de-facto standard for encryption in general. Had it not been for the mistakes made in DES, AES would have not existed at least not in its current form and for that we owe gratitude.

References

Boneh, D. (2012, April 16). *3 - 2 - The Data Encryption Standard -Cryptography-Professor Dan Boneh* [Video file]. Retrieved from https://www.youtube.com/watch?v=UgFoqxKY7cY

Grabbe, J. (n.d.). The DES Algorithm Illustrated. Retrieved from http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

History of DES. (n.d.). Retrieved from http://www.umsl.edu/~siegelj/information_theory/projects/des.netau.net/des%20history.html

Kapoor, P., Mohan, P., & Kumar, M. (2012, October 23). DES (Data Encryption Standard). Retrieved from http://priyaprakharmrigank.blogspot.qa/

Larson, M. (n.d.). What are the Differences Between DES and AES Encryption? Retrieved from http://web.townsendsecurity.com/bid/72450/What-are-the-Differences-Between-DES-and-AES-Encryption

Rouse, M. (n.d.). Data Encryption Standard (DES). Retrieved from http://searchsecurity.techtarget.com/definition/Data-Encryption-Standard

U.S. Department of Commerce, & National Institute of Standards and Technology. (1999). *Data Encryption Standard (DES)* (46-3). Retrieved from Federal Information Processing Standards Publication website: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf