A Report On The Implementation Of CBC(Cipher Block Chaining) Mode Of Operation Using DES(Data Encryption Standard) As The Underlying Pseudorandom Function

Submitted By : Amit Bhowmick(CRS2102)

MTech in Cryptology and Security (2021-23)

Indian Statistical Institute, Kolkata

Date: 27th September 2022 Project supervisor:

Place: Kolkata Dr Sabyasachi Karati

Contents

1 Abstract

In this project I did implementations of DES and CBC mode of encryption using DES as Pseudorandom Function(PRF). I assumed DES works as a PRF. Each section of this report first contain the theory of corresponding header and then the implementation details of that header done by me.

2 Introduction

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). Working of DES block cipher is as shown in the figure 1. At the encryption site, DES takes a 64-bit plaintext and creates a 64-bit ciphertext, at the decryption site, DES takes a 64-bit ciphertext and creates a 64-bit block of plaintext. The same 56-bit cipher key is used for both encryption and decryption.

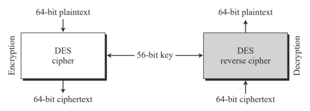


Figure 1: Encryption and decryption with DES

A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. One of popular mode of operation in literature is Cipher Block Chaining(CBC).The structure of CBC is shown in figure 2.

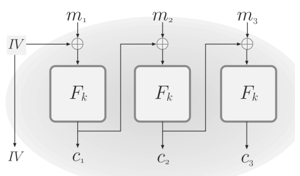


Figure 2: Cipher Block Chaining (CBC) mode

In the implementation of CBC modes of encryption using DES as the underlying PRF my code structure is as follows :

- I first took any text file. Without loss of generality I assumed this file contains 8*X byte of data, where X is the number of byte. If there is file where number of byte is not a multiple of eight, I can check the number of byte of that file using the command " ls -l filename" and can append zeros to make the byte value multiple of eight.
- On the encryption side I first run the Key-generation program which took a master key of 56 bit as random input and generate 16 round key of 48 bit each following

the key-schedule algorithm of DES. Then I run CBC mode of encryption which took 8 byte data from the text file at a time, encrypt them using DES as the underling PRF and wrote the ciphertxt in another file named "ciphertext.txt", until all data of the file read.

- On the Decryption end I first run the Key-generation program which took a master key of 56 bit as random input and generate 16 round key of 48 bit each following the key schedule algorithm of DES in reverse order. Then I run CBC mode of decryption which took 8 byte data from the ciphertext file at a time, decrypt them using inverse DES as the underling PRF and wrote the obtained plaintext in another file, until all data of the ciphertext file read.

2.1 Notation

In my implementation I used following notational convention-

- For any array a I have considered the left most bit of the binary representation of a[0] as the first bit.

3 DES Structure

Let us concentrate on encryption. later we will discuss decryption. The encryption process is made of two permutations, which we call initial and final permutations, and sixteen Feistel rounds. Each round uses a different 48-bit round key generated from the master key according to a predefined algorithm described later in this report. Figure 3 shows the structure of DES cipher at the encryption site.

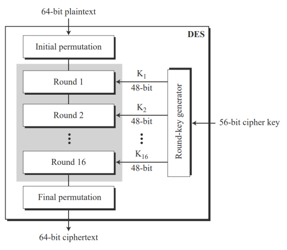


Figure 3: General structure of DES

3.1 Initial and Final Permutations

Figure 4 shows the initial and final permutations. Each of these permutations takes a 64-bit input and permutes them according to a predefined rule. I have shown only a few input ports and the corresponding output ports. These permutations are key less 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.

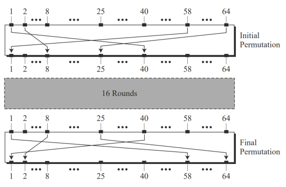
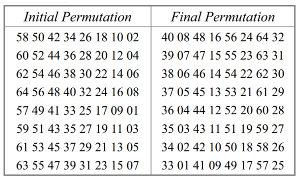


Figure 4: Initial and final permutation steps in DES

The permutation rules for these are shown in table of figure 5. Each side of the table can be thought of as a 64-element array. Note that, 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

Figure 5: Initial and final permutation tables

3.1.1 Implementation Details

//Applying initial permutation to DES input.
for(i=0;i<8;i++)

k=8;
while(k–)

int x = initial$_{m}essage_{p}ermutation[8 * i + (7 - k)] - 1;$
$j = x/8;$
$b = ((m[i] >> k)1) << (7 - xm1[j] + = b;$
$b = 0;$

$for(i = 0; i < 8; i + +)$
$m[i] = m1[i];$

//Applying final permuration to the output the DES rounds
for(i=0;i<8;i++)

k=8;
while(k–)

int x = final$_m essage_p ermutation[8 * i + (7 - k)] - 1;$
$j = x/8;$
$b = ((m[i] >> k)1) << (7 - xm1[j] + = b;$
$b = 0;$

$for(i = 0; i < 8; i + +)$
$m[i] = m1[i];$

3.2 Rounds

DES uses 16 rounds. Each round of DES is a Feistel cipher, as shown in Figure 6.

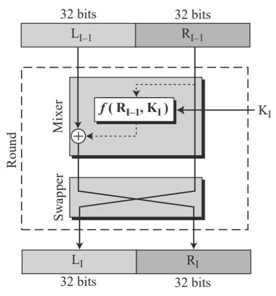


Figure 6: A round in DES (encryption site)

The round takes LI1 and RI1 from previous round (or the initial permutation box) and creates LI and RI, which go to the next round (or fi nal permutation box). Each of these rounds is invertible. The swapper is obviously invertible. It swaps the left half of the text with the right half. The mixer is invertible because of the XOR operation.

3.2.1 DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits (RI1) to produce a 32-bit output. This function is made up of four sections: an expansion D-box, a XOR (that adds key), a group of S-boxes, and a straight D-box as shown in Figure 7.

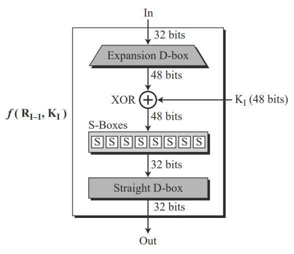


Figure 7: DES function

Expansion D-box Since RI1 is a 32-bit input and KI is a 48-bit key,we first need to expand RI1 to 48 bits. RI1 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 8 shows the input and output in the expansion permutation.

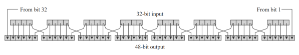
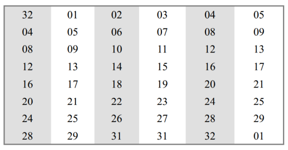


Figure 8: Expansion permutation

Although the relationship between the input and output can be defined mathematically, DES uses Table of figure 9 to define this D-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	31	31	32	01

Figure 9: Expansion D-box table

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. DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 10.

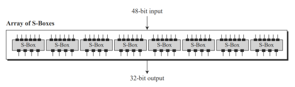


Figure 10: S-boxes

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 11.

Because each S-box has its own table, we need eight tables, for the description ofthe tables pleasse refer to reference 1, 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.

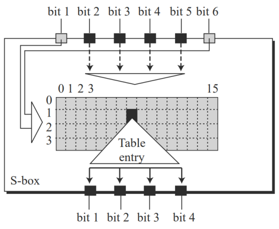
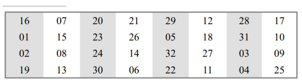


Figure 11: S-box rule

Final Permutation The last operation in the DES function is a permutation with a 32-bit input and a 32-bit output. The input/output relationship for this operation is shown in Table of figure 12 and follows the same general rule as previous tables.



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

Figure 12: Straight permutation table

3.2.2 Implementation Details

```
void f(char *r, char *answer, char *key)

//DES F fuction
char expansion[8]=0, b=0, c=0, key1[8] = 0, row = 0, col = 0, answer1[4] = 0;
inti, j = 0, bit, byte, k;


//Implementation of the expansion box starts here
for(i=0;i<4;i++)

b=(r[i]  0X0f)«1;
c=(r[i]  0Xf0)«1;
expansion[j++]+=c;
expansion[j++]+=b;
b=c=0;

for(i=0;i<8;i++)

b=((4*i-1)+32)byte=b/8;
bit=bexpansion[i]=expansion[i] | ((r[byte]»(7-bit))«5);


b=(4*i+1)byte=b/8;
bit=bexpansion[i]=expansion[i] | (r[byte]»(7-bit));
```

```
//expanding the 48 bit key in a 64 bit array to making it similar with the output of the
expansion box
byte=7;
bit=7;
for(i=5;i>=0;i–)

k=8;
while(k–)

b=key[i]  1;
key[i] »= 1;
key$_1[byte]| = (b << (7 - bit));$
$bit - -;$
$if(bit < 2)byte - -; bit = 7;$



for(i=0;i<8;i++) expansion[i] $^=key_1[i]; //XORingthekeyandtheoutputoftheexpansionbox$


//S-BOX implementation starts here.
byte=0;
for(i=0;i<8;i++)

col=(expansion[i]  0X1e)»1;
row=expansion[i]1;
expansion[i] »= 5;
row |=(expansion[i]1)«1;
if(i==0)
answer[byte] |= (S1[row*16+col])«4;

else if(i==1)
answer[byte++] |= S2[row*16+col];

else if(i==2)
answer[byte] |= (S3[row*16+col])«4;
```

else if(i==3)
answer[byte++] |= S4[row*16+col];

else if(i==4)
answer[byte] |= (S5[row*16+col])«4;

else if(i==5)
answer[byte++] |= S6[row*16+col];

else if(i==6)
answer[byte] |= (S7[row*16+col])«4;

else if(i==7)
answer[byte++] |= S8[row*16+col];

//Applying permutation to the output of the S-BOXs
for(i=0;i<4;i++)

k=7;
while(k–)

int x = right$_sub_message_permutation[8 * i + (7 - k)] - 1$;
$j = x/8$;
$b = ((answer[i] >> k)1) << (7 - xanswer1[j]+ = b$;
$b = 0$;

$for(i = 0; i < 4; i + +)answer[i] = answer1[i]$;

void Feistal(char *m, int round)

//Function for implementing the Feistal round.
int i, j;
char answer[4]=0, temp[4];

for(i=0;i<4;i++) temp[i]=m[4+i];

f(m[4], answer, K[round][0]); //Caliing the DES F function

for(i=0;i<4;i++) m[4+i]=answer[i]$^{m}[i]$;

for(i=0;i<4;i++) m[i]=temp[i];

4 Reverse DES

A simple observation of the DES structure leads me to the conclusion that the correct decryption of ciphertext(of DES encryption) will be obtained if I follow the circuit in the figure 13.

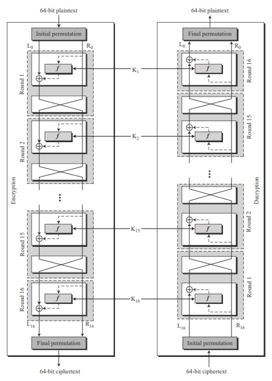


Figure 13: DES cipher and reverse cipher

4.1 Implementation Details

Implementations are here-

- DesCBCDec
- Des,h

5 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 14.

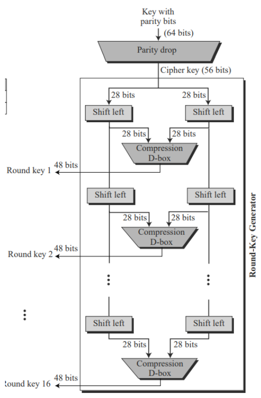
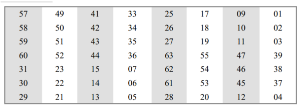


Figure 14: Key generation

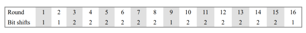
Parity Drop The preprocess before key expansion is a compression transposition step that we call parity bit drop. It drops the parity bits (bits 8, 16, 24, 32, . . . , 64) from the 64-bit key and permutes the rest of the bits according to Table of figure 15. The remaining 56-bit value is the actual cipher key which is used to generate round keys.



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

Figure 15: Parity-bit drop table

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 of figure 16 shows the number of shifts for each round.



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

Figure 16: Number of bit shifts

Compression D-box The compression D-box changes the 56 bits to 48 bits, which are used as a key for a round. The compression step is shown in Table of figure 17.

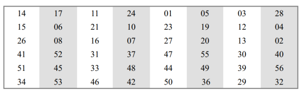
5.1 Implementation Details

Implementations are here-

- DesKeygen
- Des,h

6 Block Cipher modes of Operation

Encryption algorithms are divided into two categories based on the input type, as a block cipher and stream cipher. Block cipher is an encryption algorithm that takes a fixed size



14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Figure 17: Key-compression table

of input say b bits and produces a ciphertext of b bits again. If the input is larger than b bits it can be divided further. For different applications and uses, there are several modes of operations for a block cipher.

6.1 Electronic Code Book (ECB)

Working of ECB is as follows in the figure 18.

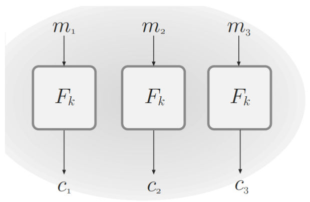


Figure 18: Electronic Code Book (ECB) mode.

6.2 Cipher Block Chaining(CBC)

Working of CBC is as follows in the figure 19.

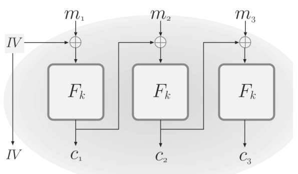


Figure 19: Cipher Block Chaining (CBC) mode.

6.3 Output Feedback Mode(OFB)

Working of OFB is as follows in the figure 20.

6.4 Implementation Details

I only did the implementation of CBC mode of encryption as a part of my midsem project. Here is my implementation-
int main()

int i, j, stop=0;
char m[8], iv[8], ch; //m holds the input of the underling DES

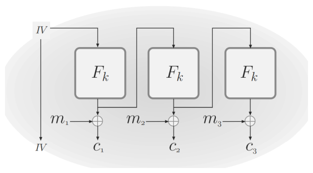


Figure 20: Output Feedback (OFB) mode.

```
ptr=fopen("message.txt", "r"); //Used to point the plaintext file.
ctr=fopen("ciphertext.txt", "w"); //Used to point the ciphertext file.


for(i=0;i<8;i++) iv[i]=rand()

for(i=0;i<8;i++) fprintf(ctr, "

//CBC mode of encryption starts here.
while(1)



//Taking the round key in a matrix form from the file previously generated by key generation algorithm.
FILE *ktr;
ktr=fopen("DesRoundKey.txt", "r");
for(i=0;i<16;i++)



for(j=0;j<6;j++)

fscanf(ktr, "

fclose(ktr);


//Taking the next 8 byte block of data from the message file which I will pass through the DES, working as the underling PRF in my CBC mode of encryption method.
for(i=0; i<8; i++)

ch = fgetc(ptr);
if(ch == EOF)
stop = 1; break;
m[i] = ch;
```

if(stop) break;

for(i=0; i<8; i++) m[i] $^{=}iv[i]$;

Des$_{E}nc(m);//callingDES$

for(i=0; i<8; i++) iv[i]=m[i];

7 References

- Data Encryption Standard (DES)
- Introduction to Modern Cryptography by Yehuda Lindell, Jonathan Katz