

CHAPTER 4

ANALYSIS AND DESIGN

4.1 Analysis

4.1.1 ECB Mode

Encrypting images using the ECB DES mode algorithm uses code created by the author himself. The DES mode ECB algorithm uses plain text with a size of 8 bytes or 64 bits. For plain text that has a size of more than 8 bytes, the method used is to divide plain text into each 8-byte section so that the encryption process is done per 8 bytes which then the results of the encryption are put back together to produce one ciphertext.

Images decryption using the ECB DES mode algorithm also uses code created by the author himself. The results of the encryption mode DES ECB algorithm produces a ciphertext measuring 16 bytes which is then decrypted to produce a plaintext with a size of 8 bytes. Just like the encryption process, if the ciphertext has a size of more than 16 bytes, the ciphertext is divided into 16 bytes then the results of the decryption are put back together.

4.1.2 CBC Mode

Encryption and decryption with CBC DES mode algorithm use a library from https://pub.dev/packages/flutter_des. Plain text can be directly encrypted without need to be separated into 8 bytes. For decryption process, enter the same key as inputted in the encryption process, otherwise the decryption results will be different from the initial plain text.

4.1.3 Encryption and Decryption Speed

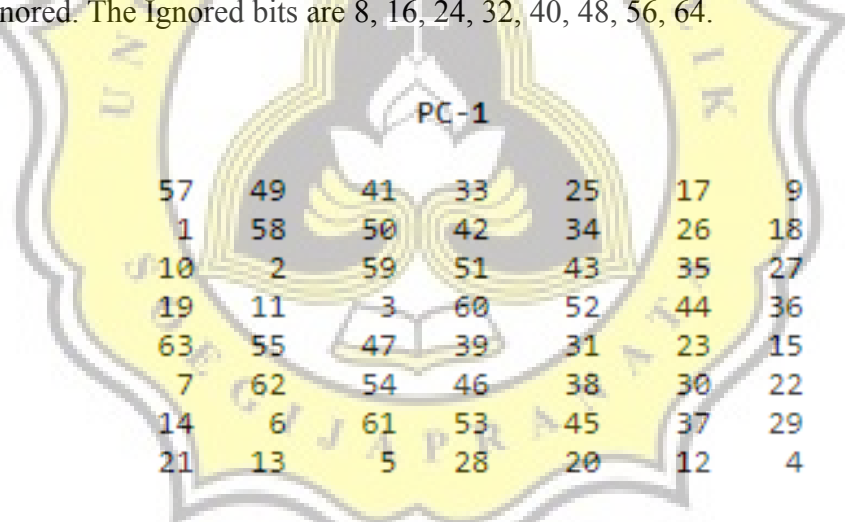
Encryption and decryption speed measurements use millisecond units. The time is calculated from the end time minus the start time.

4.2 Desain

4.2.1 Encryption

Plaintext data (messages) are encrypted into 64-bit blocks into 64-bit ciphertext data using 56-bit internal keys where the 8-bit keys will be ignored. From the way to process the plaintext, DES is categorized as a block cipher where the block cipher works by processing data in a block where several characters are combined into one block so that each process of one block produces one block of output as well.

The key given by the user is converted to 64-bit binary and compressed to 56 bits using a PC-1 compression permutation matrix while the others 8-bit keys are ignored. The Ignored bits are 8, 16, 24, 32, 40, 48, 56, 64.



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

Figure 4.1: PC-1 Table

After being compressed with PC-1 with 56 bits, the results are divided into 2 parts, with each part consisting of 28 bits and stored in C_0 and D_0 . After dividing into 2 parts, C_0 and D_0 are shifted to the left along 1 or 2 bits depending on each spin. This rotation is wrapping or round-shift, so the bit that is located in the front will move to the very back if it is moved. The rotation scheme can be seen below

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

Figure 4.2: Left Shifts Table

From the results of this shift, C_1 and D_1 are obtained. Likewise, so on to produce C_{16} and D_{16} . After that, to get the first internal key or K_1 , the bits of C_1 and D_1 need to be compressed permutations using the PC-2 matrix. So the each K_i has a length of 48 bits.

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
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 4.3: PC-2 Table

Then, plain text is permuted using the Initial Permutation (IP) table.

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

Figure 4.4: IP Table

The output of the permutation results with the IP table are divided into 2 parts, namely L_0 and R_0 where each of it consists of 32 bits. Then with the f function, it rotates 16 times with $1 \leq i \leq 16$ with 32bit data and a 48bit K_i key then X-OR. For i 1-16, count with

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} + f(R_{i-1}, K_i)$$

The results of the final block of i_{16} are L_{16} and R_{16} . For each permutation, 32 right bits are taken from the previous results and 32 left bits from the current step. In this right 32-bit step, X-OR is the left 32-bit from the previous step using the calculation f . Thus becoming

$$R_1 = L_0 + f(R_0, K_1)$$

Then encrypt the R_{i-1} 32 bit data to 48 bits with the Expansion table (E).

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	1

Figure 4.5: Expansion Table

So the results of the first 3 bits in $E(R_{i-1})$ are bits in position 32, 1 and 2 of R_{i-1} and the last 2 bits of $E(R_{i-1})$ are bits in positions 32 and 1. Furthermore, the output of $E(R_{i-1})$ is X-ORed with the K_i key.

$$K_i + E(R_{i-1})$$

Now the length of $K_i + E(R_{i-1})$ are 48 bits which can be grouped into 8 groups where each group has 6 bits. Each group is named from S1 to S8. Then each group is substituted with S-Box where in the first group is substituted with S1, the second group is substituted with S2 and so on.

S1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 4.6: S-Box Table

The next stage is the result of the calculation f , permutation is done with the P-Box table to get the value of f :

P

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

Figure 4.7: P-Box Table

$f = P(S_1(B_1), \dots, S_8(B_8))$

Furthermore, it was found that $L_2 = R_1$ and $R_2 = L_1 + f(R_1, K_2)$ and so on, which left 16 turns. At the end of round 16 you will get L_{16} and R_{16} and then reverse the two sequence blocks into 64-bit blocks.

$R_{16} L_{16}$

The application of this final permutation is done by IP-1 table.

IP⁻¹

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

Figure 4.8: IP-1 Table

The final result the permutation with IP-1 table is ciphertext in bit format and then converted to hexadecimal.

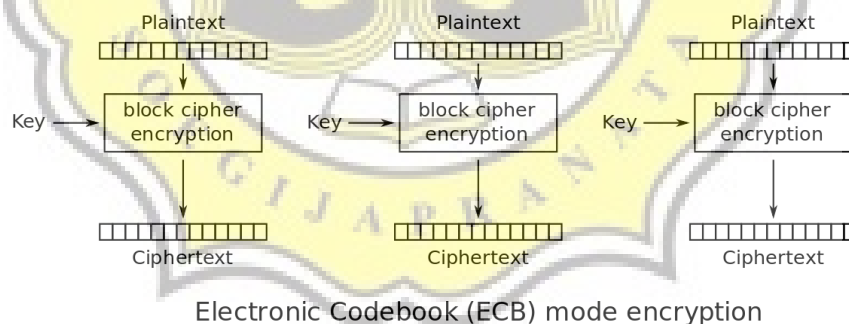


Figure 4.9: ECB Encryption Mode

For encryption of CBC DES mode algorithm, each plaintext is XORed with the previous ciphertext. To make each message unique, the use of vector initialization must be used in the first block so that each ciphertext depends on the previous one.

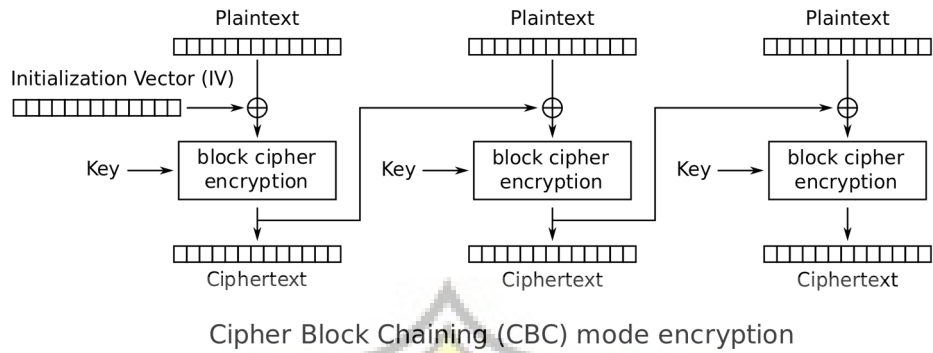


Figure 4.10: CBC Encryption Mode

To illustrate the process of image encryption, the following is an encryption mode flowchart DES ECB and CBC algorithm.

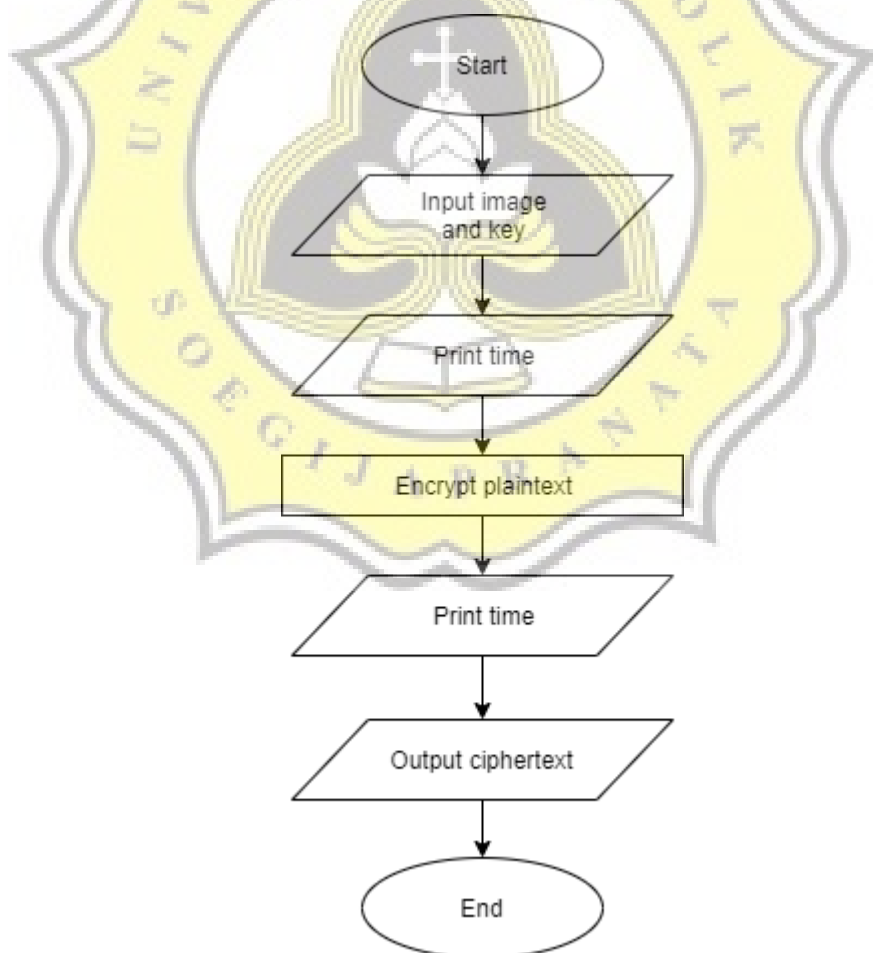


Illustration 4.1: Flowchart of Encryption Process

In the image encryption process, the first thing to do is to change the image to base64 so that it becomes a string whose length depends on the resolution of the image. The greater the image resolution, the longer the resulting string. Enter a key that is 8 bytes in size. The time is calculated when program start encrypting until the program finishes encrypting.

4.2.2 Decryption

For the decryption method, the steps to be done is only to reverse the same steps as above by reversing the order in which subkeys are applied.

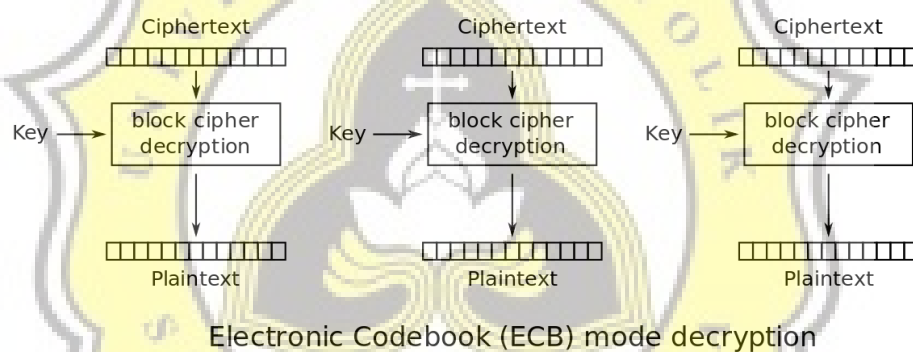
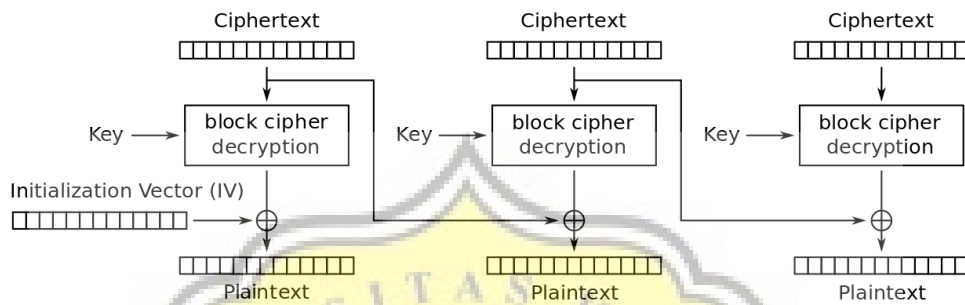


Figure 4.11: ECB Decryption Mode

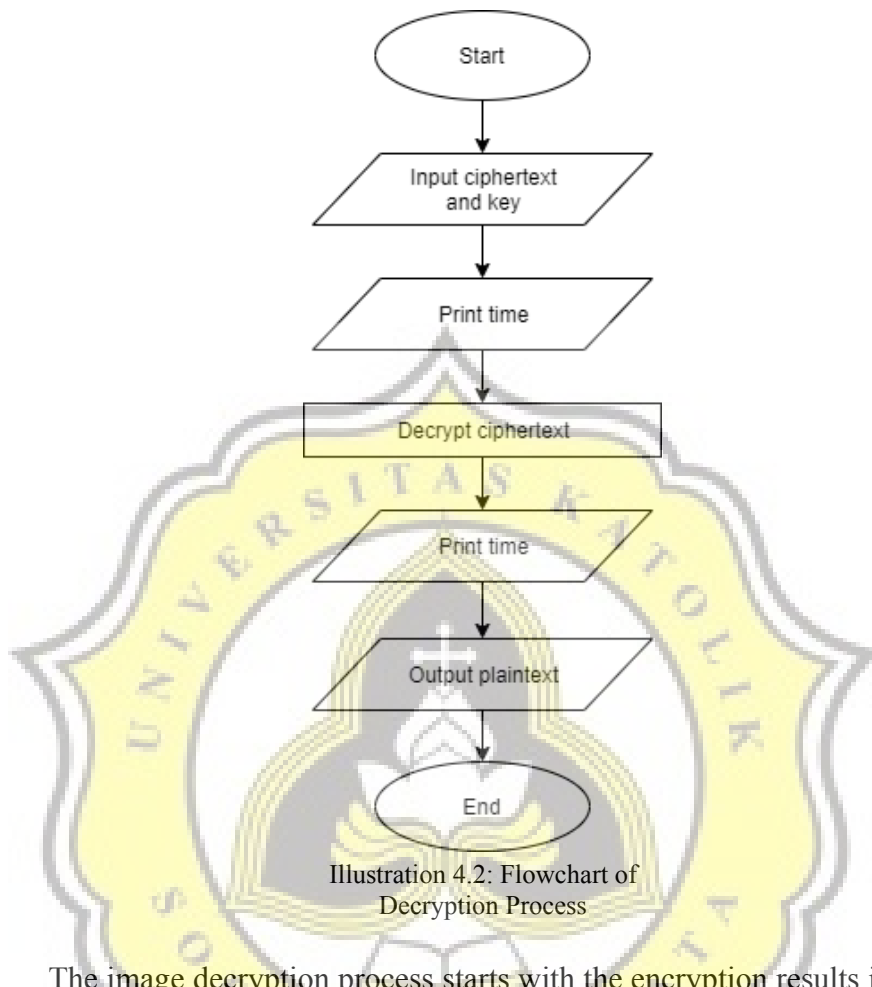
Just like the encryption process, decrypting process only needs to reverse the process of encryption with vector initialization added.



Cipher Block Chaining (CBC) mode decryption

Figure 4.12: CBC Decryption Mode

To illustrate the process of image decryption, the following is a decryption flowchart of the ECB and CBC DES mode algorithm.



The image decryption process starts with the encryption results in the form of ciphertext which is decrypted by using the same key during the encryption process. The results of the decryption will produce a plaintext where the plaintext is the result of encoding base64 from the image. After getting the plaintext, decodes base64 is used to convert the plaintext into an image.