DES 텀프로젝트
(Data Encryption Standard)

과목명	과목 6
교수	김승주 교수님
학과	사이버국방학과
학번	2021350218
이름	홍예진

<목 차>

1. 블록암호

1.1. 블록암호의 정의와 분류

- 블록암호의 정의
  - 블록암호란, 보안이 필요한 특정 정보를 암호화하는데 필요한 것으로, 정보를 특정 크기의 블록단위로 암호화와 복호화 연산을 거친다. 이 때, 각 블록의 연산에는 동일한 키를 사용하는데, 암호키 등을 블록단위로 적용하여 암호문을 만들어 암호화하는 것이 블록암호이다. 특정 메시지에서 동일한 블록이 동일한 암호문으로 암호화되는 것을 막기 위해 이전 단계의 암호블록의 암호문을 그 다음 블록에 순차적으로 적용한다.
- 블록암호의 분류
  - 블록암호에는 페스탈(Feistel)구조와 SPN구조가 있다. 페스탈구조의 장점은 역함수가 필요가 없지만, 암호화를 구현할 때 스왑(SWAP)단계에서 연산량이 많다는 단점이 있다. 또한 암호화 과정에서 사용되는 라운드(round)함수를 안전하게 설계해야 한다. SPN구조는 비트의 이동이 없이 한번에 암호화, 복호화가 가능하다. 이 때문에 페스탈구조의 블록암호에 비해 더욱 효율적으로 설계가 가능하다. 하지만 암호화, 복호화의 과정에서 페스탈구조와 달리 역함수가 필요하다. SPN구조에 해당하는 블록암호로는 AES가 있다.

2. DES

2.1. DES의 정의와 분류

- 데이터 암호화 표준(Data Encryption Standard), 즉 DES는 64 bits의 블록 크기와(64bit의 길이를 갖는 평문) 56 bit의 키 길이를 갖는 암호 알고리즘으로, 블록암호의 일종이다. DES는 1977년 미국 정부의 암호 표준으로 발표되었다. 64 bit의 키가 사용되지만 이 중 56 bit는 실질적인 내부키이고 남은 8bit는 거사용 비트(Parity Bits)이다. DES는 페스탈구조와 SPN구조로 분류되는 블록암호에서 페스탈(Feistel) 구조에 해당한다.

2.2. DES 구조

- 데이터 암호화 표준(DES)의 구조는 Feistel의 형태로, 16회의 라운드(round)를 거친다. 또한 DES는 64bit의 블록길이와 56bit의 키 길이를 가지며. 각 라운드에서는 48bit의 키가 사용된다. 실제 key암호화에 사용되는 키의 사이즈는 56bit인데 8bit는 앞서 언급한 바와 같이 parity bit로 사용되므로 DES에서는 64bit의 키를 사용한다. 페스탈(Feistel) 구조는 치환(Substitution)과 순열(Permutation)을 교차하여 암호화를 수행한다. 페스탈(Feistel)구조를 갖는 DES는 데이터를 좌(Left)와 우(Right) 두 부분으로 나누어 비선형 변환의 연산을 거친다는 특징을 갖는다.

2.3. DES의 적용(DES가 적용되는 과정, 작동과정 예시)

- DES의 적용과정은 다음과 같다. 우선 64-bit의 평문을 한 블록단위로 받는다. 이러한 64-bit의 블록이 IP(Initial Permutation) table을 거쳐 순서가 뒤섞이게 된다. IP를 거친 후에는 Left와 Right로 각각 32bit씩 나눠지고 16개의 라운드를 거치게 된다. 한 라운드에서는 64-bit의 키가 PC-1, PC-2table을 거치면서 48-bit로 압축되고, Right의 32-bit는 48-bit로 확장되는데, 이 둘을 XOR 연산을 거쳐 48-bit를 출력한다. 이렇게 출력된 48-bit는 8개의 S-box를 거쳐 다시 32-bit로 변환된다. S-box에서 출력된 32-bit는 P-box(permutation) table을 거치며 또 다시 뒤섞이는 과정을 거친다. P-box를 거친 32-bit와 Left를 XOR연산을 하면 다음 라운드에 들어갈 Right의 32-bit가 완성되고, 다음 라운드의 Left는 이전 라운드의 Right 32-bit가 그대로 사용되는데, 이를 16번 반복한다. 즉, 16번의 라운드를 거치는 것이다. 마지막 라운드에서는 FP(Final permutation)을 거치고, 최종적으로 64-bit의 암호문이 완성된다.

2.4. Permutation

- Initial Permutation(IP) (IP의 현재 형태와 현재 형태를 갖게된 이유)

- 순열, 즉, Permutation은 페스탈 구조를 가지는 DES에 사용된다. Permutation은 IP(Initial Permutation)과 FP(Final Permutation)이 있다. 이때, FP는 과 같다. IP와 FP에서는 아직 암호화가 일어나지는 않는다. IP와 FP는 Permutation Table에 명시된 바에 따라 bit교환이 일어난다. 각 비트들은 Permutation table을 거치면 스왑(swap)이 일어난다. IP와 FP의 permutation table은 다음과 같다.

Initial Permutation(IP)	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

- Final Permutation(FP 또는)

- Fianl Permutation, 즉 FP 또는 는 마지막 라운드인 16번째 라운드에서 사용되는 permutation으로, initial permutation(IP)의 역함수라고 할 수 있다.
  64bit의 평문이 IP를 거쳐 Initial Permutation에 따라 스왑(swap)되면, 첫 번째 Round함수에 들어가기 전 왼쪽(Left) 32 bit와 오른쪽(Right) 32 bit로 나뉘게 된다. 즉, 각각 32bit의 L과 R로 표현할 수 있다.

2.5. Round 함수

- Round 함수
  - Permutation에서는 DES의 암호화가 일어나지 않는다고 앞서 언급했는데, DES의 암호화는 이 Round함수 단계에서 일어난다. DES에서 Round 함수는 총 16번의 라운드를 거친다. 64비트의 평문이 IP를 거친 후 L 32 bit와 R 32 bit로 나뉘면, 평문 L과 R의 32 bit는 확장, S-box, P-box, Xor 연산을 거친 뒤 다음 라운드의 함수로 들어간다. 즉, i번째 라운드를 거친 평문 L과 R을 갱 갱라고 하고, 그 전 단계의 L과 R을 갱 갱라고 하면,

갱 갱

갱 갱 ⊕갂 갱 갱

이라고 할 수 있다. 이 때, f는 라운드 함수를 의미하고, K는 48 bit의 Key를 의미한다.(Subkey) 갱는 I번째 라운드에서 사용되는 subkey이라고 할 수 있다.
이제 위 수식의 계산과정을 살펴보자.

- 키 변환(Key Transformation)
  - 앞서 DES의 정의에서 DES는 64 bit의 키 중 56 bit의 실질적인 내부키를 갖는다고 했는데, round함수에서 이 키가 사용된다. 참고로 64 bit에서 실질적인 내부키인 56-bit 외에 8-bit는 오류검증에 사용되는 parity bit이다. parity bit는 8, 16, 24, 32, 40, 48, 56, 64 번째 비트가 사용된다. 그런데 앞서 subkey K는 48 bit의 Key라고 했다. 이는 라운드함수가 본격적으로 실행되기 전, 키변환(Key Transformation)이 일어나기 때문이다. 이는 매 라운드마다 실행된다. 56 bit의 키를 48bit로 만들기 위해서는 우선 56 bit를 28 bit 씩 나눠진다. 이에 대해 bit의 위치(Bit-Position)가 달라지는 Shifting Process가 일어나는데, 매 라운드마다 실행된다고 했으므로, 16번 일어난다. 이때, 1,2,9,16 번째 subkey에서는 하나의 bit 포지션만 바뀌고(left shift가 1번) 나머지 라운드들에서는 두 개의 bit의 포지션이 바뀐다.(left shift가 2번)

    한 라운드안에서 위 과정을 마치면, 압축 치환(compression permutation)이 일어난다. 즉, 56 bit를 48 bit로 압축하는 것이다. 이는 기존의 56개의 비트에서 9, 18, 22, 25, 35, 38, 43, 54 번째 비트를 없앰으로서 만들 수 있다. 예를 들면, 다음과 같다. 아래 표에서 압축된 48 bit의 키를 확인할 수 있다.

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

따라서 키변환 과정에서 64-bit에서 56-bit로 한번, 56-bit에서 48-bit에서 한번 압축이 일어난다고 할 수 있는데, 64-bit에서 56-bit가 되는 과정에서 PC-1 table이, 56-bit에서 48-bit로 압축될 때 PC-2 table이 사용된다.

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

PC-1 table

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

PC-2 table

(Shifting Process는 다음과 같이 나타낼 수 있다.)

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

그림 | Shifting Process

또한 키 변환과정을 도식화해서 나타내면 다음 그림과 같다.

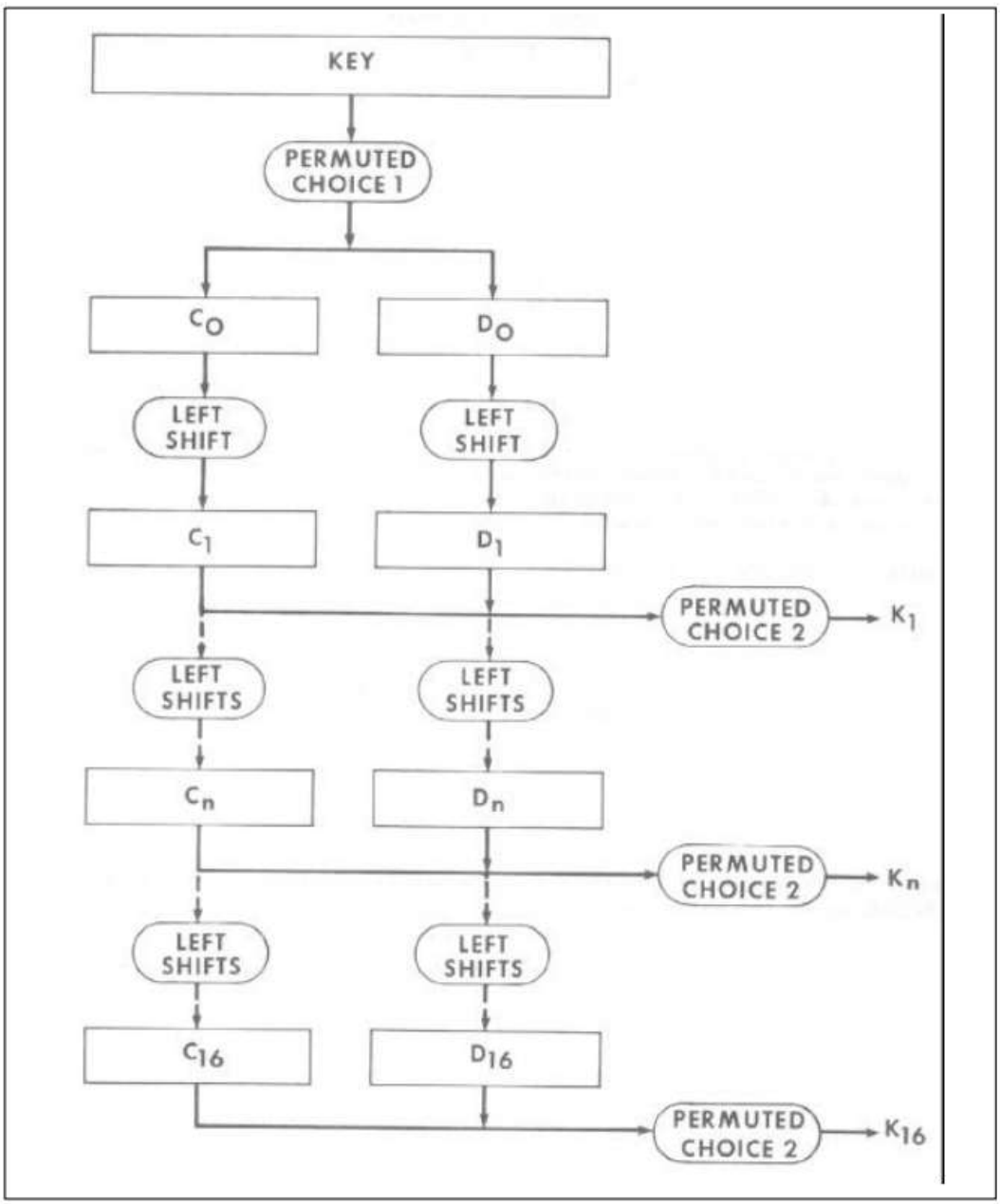


그림 | Key schedule calculation

- 확장 치환(Expansion Permutation)과 XOR
  - 32-bit의 R평문을 48-bit로 확장해야 다음 연산을 계속할 수 있다. 32-bit의 R평문은 한 block당 4-bit로 이뤄져 있다. 즉, 4-bit인 8개의 block으로 이루어져 있는 것이다. 이제 32-bit를 48-bit로 확장하려면 한 block당 4-bit를 6-bit로 변환해 주면 된다. 그러면 총 6-bit인 8개의 block이 만들어지므로 48-bit로 확장이 됨을 알 수 있다. 4-bit를 6-bit로 변환하는 것은 어렵지 않다. 첫 번째 block의

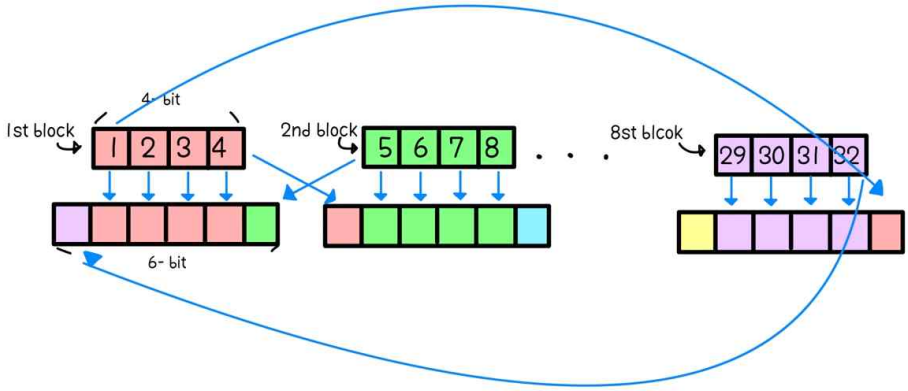


그림 | Plain text의 Expansion(32 bit->48 bit)

비트부터 순서대로 편의를 위해 1, 2, 3, 4,..., 29, 30, 31, 32, 까지 숫자를 붙이면, 1~4, 5~8, 9~12, 13~16, 17~20, 21~24, 25~28, 29~32 번째 비트들이 각각 같은 block임을 알 수 있다. 6-bit로 확장될 때는 각 블록이 6bit의 공간이 되고, 양끝을 제외한 가운데 4-bit에는 기존의 4개의 bit가 그대로 위치한다. 오른쪽 끝에는 다음 block의 첫 번째 bit가 위치하고, 왼쪽 끝에는 이전 block의 마지막 bit가 위치한다. 첫 번째 확장된 block의 경우 첫 번째 자리에는 맨 마지막 block의 마지막 비트가 위치하고, 8번째 확장된 block의 가장 오른쪽 자리에는 첫 번째 block의 첫 번째 bit가 위치하게 된다. 이를 그림으로 나타내면 다음과 같다.

즉, 한 블록당 4bit를 가진 8개의 블록을(4 bit x 8 = 32 bit) 한 블록당 6-bit를 가지는 8개의 블록(6 bit x 8 = 48 bit)으로 변환하는 것이다. 이렇게 평문의 확장치환을 모두 마쳤다면, 다음 단계에서는 키변환을 마친 48-bit의 키와 확장치환된 48-bit의 Right는 XOR연산을 거치게 된다. L'을 다음라운드의 Left, R'을 다음 라운드의 Right, f(R,K)를 Right 32-bit와 Key가 사용되는 함수라고 하면 ′ ′ ⊕갂 와 같이 표현될 수 있고, KS를 ≤ 걗 ≤ 인 n에 대해 64bit의 KEY를 input으로 받아 48-bit의 Key 걗을 출력해내는 함수라고 하면, 걗 걗 ′ 라고 할 수 있다. 즉, 위의 식을 n을 포함해 다시 쓰면, 걗 걗 라고 할 수 있다. DES는 16개의 라운드를 거치므로, KS는 걗 걗 ⊕갂 걗 걗 알고리즘에서 Key인 걗도 16개를 만든다고 할 수 있다.

- S box Substitution
  - 확장 치환된 48-bit의 R과 48-bit로 압축된 키의 XOR을 통해 얻어진 48-bit의 출력은 S box에서 다시 32-bit로 변환된다. 라운드 함수는 16번 실행되기 때문에

S-box도 16번 사용된다. S-box는 앞서 설명한 IP와 같이 주어진 표(table)에 따라 변환된다. S-box는 총 8개인데, 48-bit를 32-bit로 다시 변환해야 하므로, 각 8개의 S-box는 각각의 블록에 적용된다고 할 수 있다. 예를 들어, 한 블록당 6bit이므로, 첫 번째 블록에는 S1, 2번째 블록에는 S2,...8번째 블록에는 S8을 적용하는 식이다. 이렇게 되면 6bit*8= 48bit이므로 모든 블록, 즉 모든 bit에 대해 적용되어 32bit로 줄일 수 있다. S-box는 다음 표와 같다.

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

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

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

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

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

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

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

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

S-box가 적용되는 과정은 다음과 같다. 6-bit의 각 block에서 끝자리의 두 bit와 가운데 네 자리의 bit를 각각 따르면 된다. 끝에 두 bit는 00, 01, 10, 11 중 한가지의 형태를 띌 것인데, 이를 10진수로 표현하면 0,1,2,3이라고 할 수 있다. 그리고 가운데 4개의 bit역시 0000, 0001, ..., 1110, 1111총 16가지의 경우중 한가지를 갖게 될 것인데, 이 역시 10진법에 따르면 0~15의 수로 표현된다. 따라서 끝자리 두 bit는 열, 가운데 네 bit는 행을 나타낸다. 즉, 끝에 두 bit가 00이면 첫 번째 줄, 01이면 두 번째 줄, 10이면 세 번째 줄, 11이면 네 번째 줄이고, 가운데 4비트가 0000이면 첫 번째 행, 0001이면 두 번째 행, ..., 1110이면 14번째 행, 1111이면 15번째 행에 해당된다. 따라서 두 가지를 모두 만족하는 숫자에 대응되게 된다. 예를 들어, 이전 단계에서 출력된 48-bit중 011011인 한 block이 있다고 하면, 양 끝의 두 bit와 가운데 네 bit의 값을 이용해 정의된 표를 따라 4-bit로 변환되는 것이다. 따라서 주어진 표를 이용해 변환되는 값을 찾으면 된다. 이때 라운드에서 S5을 따라야 한다고 할 때, 양 끝의 비트는 01 이고, 가운데 네 bit는 1101이므로, 가로로는 앞에서 13번째, 세로로는 두 번째 줄의 숫자인 9(1001)로 변환할 수 있는 것이다. (표는 2진법으로 표현한 15개의 숫자 0000~1111를 10진법으로 0~15까지 표현한 것이다.) 또 다른 예를 들어보자면, sbox를 거치기 전 한 블록의 여섯 bit가 010011이라면, 양끝 bit는 01로, 십진수로 표현하면 1, 가운데 네 bit는 1001로, 십진수로 표현하면 9이므로, 이때 S1을 따른다고 가정하면, 표에 따라 6, 즉, 0110의 4-bit를 출력하게 되는 것이다.

S_1

Row No.	Column Number															
	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

그림 | S1 box 적용

- P box (Permutation)

• S box의 출력값은 P box의 입력으로 들어오게 된다. P box도 앞서 IP, S box와 마찬가지로 table을 가지는데 table은 다음과 같다.

P Box Table (Permutation Function)															
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

위 표에 따라 마찬가지로 Permutation연산을 진행하면 32-bit의 출력값이 나오게 된다. (P box에 의해 32-bit가 섞인다.) 이 때 P-box는 앞서 IP와 같이 bit를 뒤섞어 주는 역할을 한다고 할 수 있다.

- XOR 과 SWAP

• 이제 Round함수를 거친 평문 R의 32-bit와 L의 32-bit를 최종적으로 XOR연산을 취해주면, 이는 다음 라운드의 R이 된다. 그리고 다음 라운드의 L은 앞서 말한 것처럼 이전 라운드의 R이 그대로 들어가게 된다. XOR연산은 bit가 같은 값을 가지면 0, 다른 값을 가지면 1을 출력하는 연산이라고 할 수 있다. 예를 들어, 라운드 함수를 거친 32bit의 R과 32bit의 L을 XOR연산을 할 때, 예를 들어 R의 앞 네개의 bit가 0110, L의 앞 네 개의 bit가 1011이라고 하면, 순서대로 다름, 다름, 같음, 다름이므로 XOR연산을 거치면 1101이 출력되는 것이다. 위 과정을 모두 마치면 한 라운드가 끝나고, 위와 같은 과정을 16번 반복하면 되는데, 마지막 라운드인 round 16에서는 앞서 언급한 최종순열 즉 Final Permnutation(FP 또는)을 수행한다. FP, 즉 는 말그대로 IP의 역이기 때문에 IP에서는 평문의 58번째 bit가 첫 번째 자리로, 평문의 50번째 bit가 두 번째 자리로,...,,평문의 40번째 bit가 28번째로,..., 평문의 7번째 bit가 64번째에 배치되었다면, FP에서는 그 역으로 배치가 된다. (역함수라고 생각하면 된다.) 그리고 이 때는 예외적으로 앞서 Left, Right 순서였던 것과 달리 Right, Left 순으로 표시된다. round 16 까지 마치게 되면 최종적으로 64-bit의 암호화된 블록을 얻게 된다. 지금까지 설명한 DES 알고리즘의 작동과정을 도식화하면 다음 그림과 같다. 암호화과정에서 라고 정의했듯, 복호화(Deciphering) 과정에서는 반대로 , 즉, 라고 할 수 있다. 암호화함수 f, 즉, 라운드함수라고 할 수 있는 함수 f를 도식화해 나타낸 그림은 다음과 같다.

2.6. DES의 효과

• 대칭 키 암호 알고리즘에 속하는 DES는 사전에 암호화되었던 문서를 복호화하는 용도로 사용되기도 한다. 대칭키 암호는 연속적인 비트를 입력받고 이에 대응하는 암호화된 비트를 출력하는 스트림 암호 방식과 특정 단위의 블록을 입력받아 이에 대응하는 암호화된 블록을 출력하는 방식인 블록암호 방식이 있는데, DES는 블록암호에 해당한다. 대칭키 암호 알고리즘은 치환이나 전치같은 간단한 내부구조를 갖는

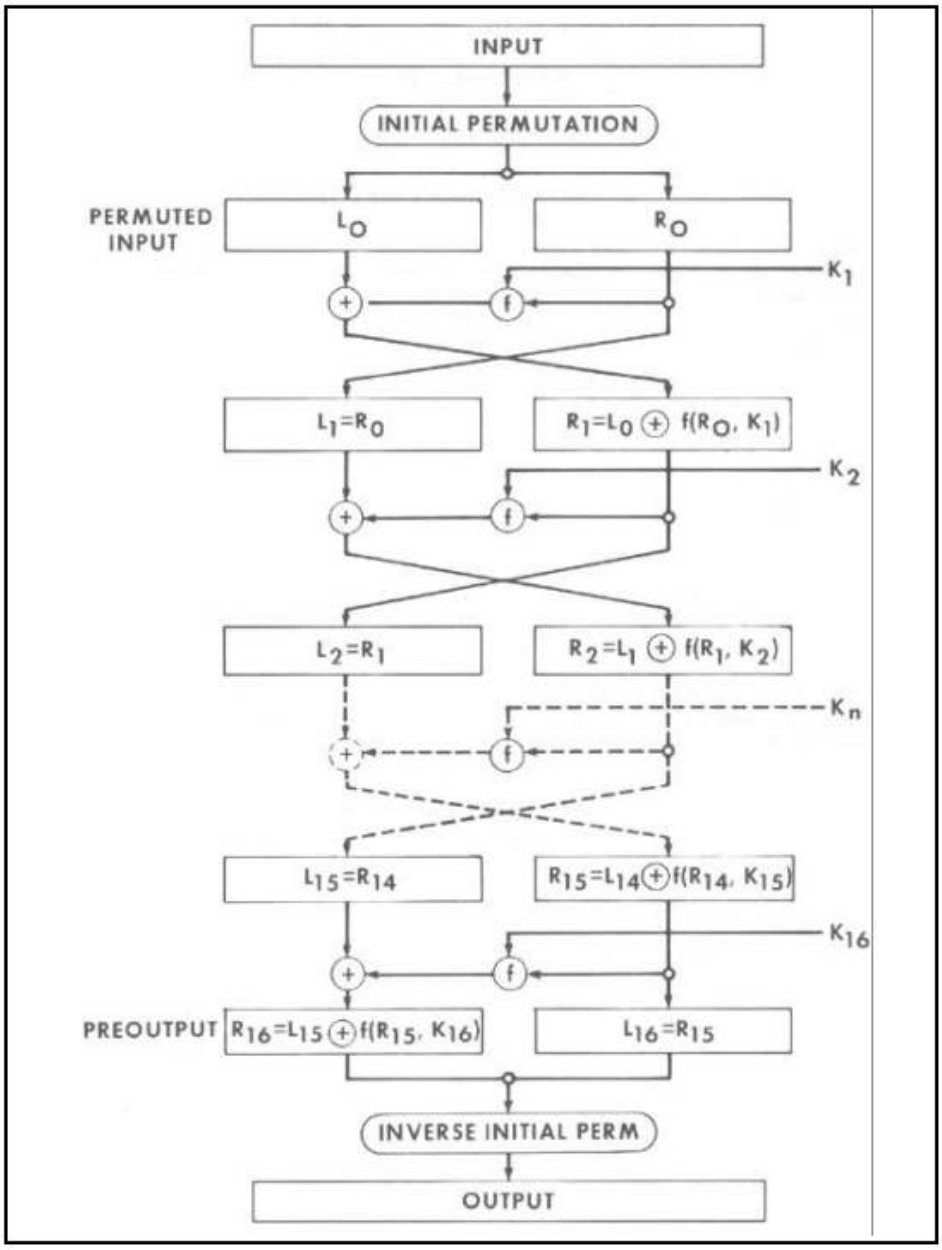


그림 | Enciphering computation

알고리즘을 가져 개발이 쉽고 시스템 개발환경에 용이하며 대칭키를 갖기 때문에 비대칭키를 갖는 알고리즘에 비해 컴퓨터 시스템에서 빠르게 동작한다. (암호화, 복호화 속도가 더 빠르기 때문) 이러한 DES는 중요한 데이터를 전송하거나 저장할 때, 기밀성을 유지하면서 데이터를 보호할 수 있다. 또 암호화 과정에서 사용되는 키와 관련해 보안성이 유지된다. 공격기술 중 하나로는 선형공격(linear cryptanalysis)이 있는데, 선형공격은 암호공격의 종류 중 하나이다. 선형공격은 암호화과정의 선형관계식 찾기를 목표로 하는데, 주로 블록암호를 공격할 때 많이 쓰이는 방법이다. DES의 경우 평문P, 암호화 키K, 암호문 C가 나온다고 할 때, 각각

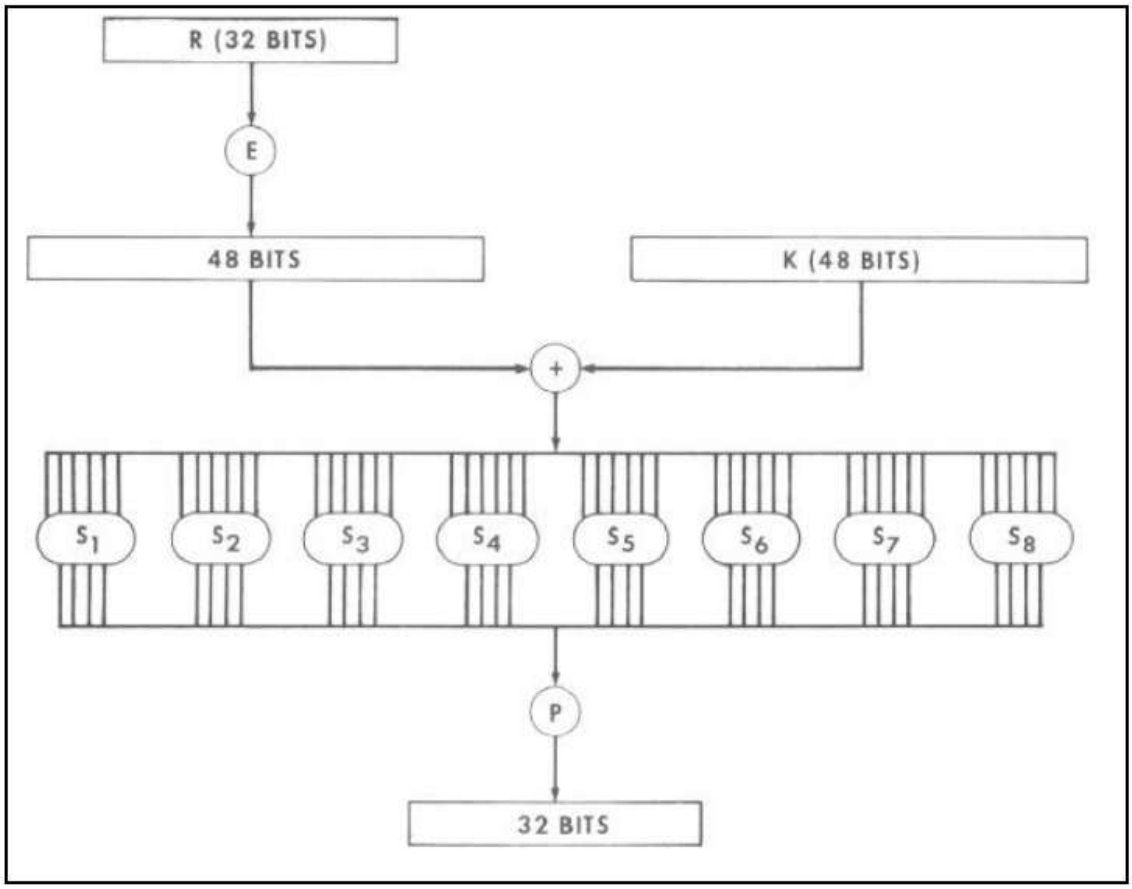


그림 | Calculation of f(R,K)

의 P,K,C가 어떤 관계를 가지는지 또는 관계식이 성립하는 확률이 어떤지를 알아내려 한다. 보통 관계식이 성립하는 확률이 1/2이면 암호화함수가 이상적이라고 할 수 있지만, 이 확률은 함수마다 다르다. DES에서는 대입(substitution)과 혼합(permutation)이 일어나는데, permutation의 과정은 선형적이고, 라운드함수 내에서 사용되는 S-box의 경우 대입과정에서 비선형적인 변환과정을 거친다고 할 수 있다. 따라서 선형공격에서는 비선형적 변환과정을 거치는 Sbox에서 선형적으로 연산을 찾는 것을 시도한다. 이를 찾으면, 이를 통해 chosen-plaintext라는 공격을 할 수 있다. 즉 DES의 S-box는 암호화의 과정에서 다른 과정과 달리 비선형성을 띄기 때문에 선형공격에 강하다고 할 수 있는 것이다. 이러한 비선형성을 띄는 과정이 없었다면 선형대수적인 방법의 공격에 취약했을 것이기 때문이다. 또한 DES에는 Sbox외에도, 확장(expansion), 비트 혼합(permutation)등 여러 함수가 사용되어 입력비트가 조금만 달라져도 출력에는 그 이상의 영향을 끼치는 확산(diffusion)의 효과도 가진다.

2.7. DES의 취약점

- 암호키에서 선택된 몇몇 비트로 각 라운드(round)의 키로 구성되기 때문에 암호키가 0이면 각 라운드의 키는 0이 된다. 이는 복호화과정도 마찬가지다. 따라서 반대

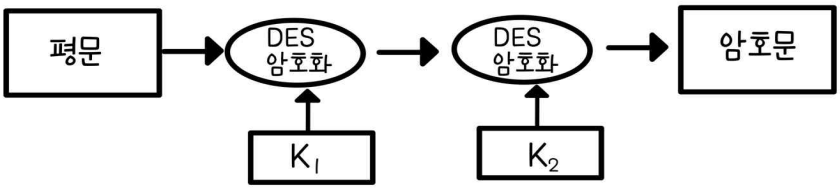
방향으로 n번째 키(≤ 잔 ≤)을 이용해 16라운드를 거치고 의 계산을 거치면(는 의 역함수/역 연산) 처음 평문을 얻을 수 있다는 단점이 있다. 또한 DES의 키의 크기가 56bit으로 길지 않다보니, 앞서 말한바와 같이 평문을 암호문을 풀 수 있다. 이 부분은 기술이 발전하면서 공격기술도 향상되었고, 자연스럽게 DES의 취약점이 되었다. 이를 보완하기 위해 키의 길이를 늘린 다중 DES가 나타나게 되었다. 키의 길이를 늘리면 공격할 때 경우의수도 늘어나기 때문에 56bit의 키의 크기를 갖는 DES보다 안전성이 높아진다고 할 수 있다. 정보를 교환하는 사람끼리 동일한 키를 공유해야 하기 때문에 키의 교환과 관리가 어렵다는 점도 DES의 단점 중 하나이다. 즉, DES는 많이 사용되었었지만, 현재의 공격기술로 비추어봤을 때, 현재는 사용하기에 키 길이가 비교적 짧아 안전하지 않다는 것이다. 또한 DES는 한 블록당 64-bit의 평문을 받아 암호화하는데, 이는 현재 비교적 블록의 크기가 작다고 할 수 있다. 즉 블록의 크기가 작기 때문에 키 하나당 암호화할 수 있는 문서의 양이 적다는 취약점도 갖는다.

2.8. DES의 변형과 차이점

- 2중 DES

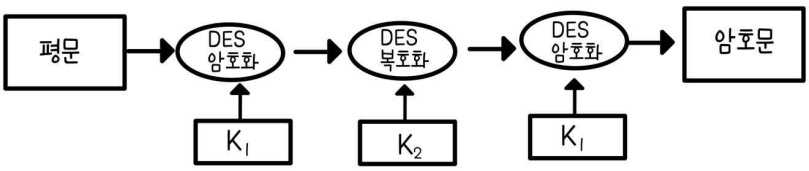
• 키의 길이가 짧은 single DES(단순 DES)를 보완하기 위해 키의 길이를 2배 늘려 112-bit의 키를 사용하는 방법이 2중 DES이다. 2중 DES는 단순 DES에 비해 연산량이 2배이기 때문에 비교적 효율성이 떨어지기도 한다. 그런데 아래 설명되어 있는 3중DES와 달리 2중 DES는 현재 잘 쓰이지 않는다. 그 이유는 2중DES임에도 불구하고 단순 DES를 공격하는 방식과 큰 차이가 없는 방법으로 2중DES를 공격할 수 있기 때문이다. 이 2중 DES를 공격하는 방법을 Meet in the Middle Attack 라고 한다. 2중 DES는 평문을 키 으로 DES암호화 하고, 또 다른 키 에 대해 한번 더 DES암호화해 암호문을 얻는 방식이다. 이에 대해 Meet in the Middle Attack은 과 를 찾아내는데, 먼저 평문을 키 으로 DES암호화 한 값과 키 을 포함하는 표에 대해 미리 연산을 한다.(표는 256크기를 가진다) 그 후 평문을 으로 DES암호화한 값으로 표를 배열하고 이 표를 통해서 다른 키 를 이용해 암호문 C를 복호화한 값과 같아질 때까지 C의 복호화를 반복하면 된다. 다시 말해서, 평문을 키 으로 암호화한 값과 암호문을 복호화한 값이 같을 때를 찾으면 그 때의 두 키의 값 과 를 찾으면 이를 2중 DES의 키 쌍이라고 할 수 있는데, 2중 DES에 대한 공격, 즉, Meet in the Middle Attack은 크기의 표에 대한 연산을 미리 해야 하지만 한번만 수행하고 2DES를 여러번 공격할 수 있다. 이 표를 만들고 나면 과 를 찾은 계산만 하면 되는데 이 작업의 기댓값은 이다. 그런데 이는 single DES(단순 DES)를 공격할 때 사용하는 방법의 작업량과 동일하기 때문에 2중 DES가 단순 DES보다 효율적이고 안전하다고 하기는 어렵다. 따라서 2중 DES는 현재 잘 사용되

지 않는다.



2중 DES

- 3중 DES
  - 3중 DES는 DES를 3번 수행한 암호 알고리즘이라고 할 수 있다. 3번 반복해서 수행하기 때문에 2중 DES의 공격방법인 Meet in the Middle Attack 방법을 사용하기에는 무리가 있을 것이다. (실행하려 한다면 연산량이 너무 많아지므로)
    3중 DES가 2중 DES와 가지는 차이점은 다음과 같다. 2중 DES는 두 개의 키를 이용해 암호화-암호화를 해서 암호문을 얻었다면, 3중 DES는 일반적으로 서로 다른 키를 두 개 가지되, 암호화-복호화-암호화를 통해 암호문을 얻는다. 따라서 라운드는 16*3=48 round가 된다. 또한 키 $_1$과 $_2$가 있다고 하면, 키 $_1$으로 암호화하고, $_2$로 복호화, $_1$으로 암호화한다. 3중DES가 암호화-복호화-암호화 방식을 사용하는 이유는 단순 DES와의 호환성 때문이다. (3중 DES에서 서로 다른 키 2개가 같은 키라면 단순 DES와 같은 형태가 된다. 3중 DES가 3개의 키가 아닌 2개의 서로 다른 키만 사용하는 이유는 112-bit(56*2)의 키만으로도 높은 안전성을 갖기 때문이다. 하지만 더욱 강력한 보안을 원한다면 3개의 서로 다른 키를 사용하는 것도 가능하다. 이 때는 키 $_1$으로 암호화후 $_2$로 복호화하고, $_3$으로 암호화하면 된다.



3중 DES

3. 표준문서

3.1. 표준문서의 구성

- DES표준문서는 서문(Foreword)과 초록(Abstract)을 통해 NIST에서 DES를 국가표준으로 정한것과, 정보를 주고받는 과정에서 DES, TDEA등을 통해 보안을 강화할 수 있음을 명시하고 있다. 서문과 초록이 끝 난 뒤 6pg부터는 이름(Name of Standard), Category, DES에서 키의 길이, parity bit, TDEA는 3개의 DES키로 이루어져 있는 등 DES에 대한 간략한 설명부터 DES의 적용, 실행등 전반적인 DES의 배경지식에 대해 다루고 있다. 본격적인 DES알고리즘에 대한 설명은 표준문서의

13pg부터 확인할 수 있다. DES알고리즘에 대해 크게 Introduction(도입), Enciphering(암호화), Deciphering(복호화), The Cipher Function f(암호화 함수 f), 로 나누어 ////DES가 어떠한 용도로 사용되는지, 어떤 기관에서 만들어졌는지 등부터 시작해 DES에 필요한 암호화 알고리즘, DES의 구성 (DES에서 암호화에 필요한 요소 등), 데이터를 암호화할 때 필요한 조건 등으로 구성되어 있다. 또한 이후 본격적으로 DES의 작동원리에 대해 설명하고 있다. 데이터암호화 표준의 알고리즘 작동과정에서 사용되는 키, 함수, 치환, 축소, 대체, 확장, XOR연산, 라운드함수에서 일어나는 과정 그리고 DES의 취약점과 이를 보완하기 위한 3중 DES(Triple Data Encryption)과 3중알고리즘의 작동원리까지 설명하고 있다.

3.2. 표준문서의 내용

- DES는 중요한 데이터를 전송하거나 저장할 때 데이터를 보호하는 용도로 사용된다. DES는 암호화 알고리즘을 통해 데이터를 암호화하고, 이진수를 기반으로 암호화를 진행한다. RAM, PROM 등 DES가 사용되는 사례나 기술에 대해 설명하며 DES 알고리즘의 구현은 환경이나 기술 등 여러 요인에 따라 조금씩 달라질 수 있다. 또한 키와 관련해서는 통신 중 송신자와 수신자가 동시에 키를 갖고 있어야 함 또한 언급하고 있다.
- 표준문서의 Introduction 부분에서는 DES의 키는 64bit인데 이 중 56 bit가 랜덤하게 생성되어 알고리즘에 의해 직접 사용되며, 이 외의 나머지 8bit는 오류를 탐지할 때 사용된다는 내용이 포함되어 있다. (parity bit) 이러한 키의 변환을 설명하는 Enciphering 파트에서는 키 변환이 매 round마다 일어나고, 이 과정에서 PC-1, PC-2 table을 이용해 압축치환을 거쳐 64-bit에서 48-bit의 키가 되는 과정을 설명한다. 또한 64-bit의 평문이 Initial Permutation(IP)를 거쳐 Left와 Right로 32-bit 씩 나누어져 추후 연산을 거치며, 이 과정에서 IP의 역인 도 사용된다는 내용을 다룬다. 문서를 통해 총 16 라운드가 반복되므로 각 라운드를 거치며 다음 Left, Right가 이전 Left, Right와 어떤 관계인지를 알 수 있다.(어떤 계산을 거쳐 다음 Left 와 Right가 생성되는지) 그리고 앞서 언급된 압축치환된 48-bit의 키가 이 계산 과정에서 사용됨을 알 수 있다.(XOR연산)
- 암호화의 역연산인, 복호화에 대해 설명하는 Deciphering파트에서는 사전출력에 적용된 순열 IP의 역순인 가 사용되며, 해독하기 위해서는 동일한 알고리즘을 사용해 계산을 반복되며, 같은 키가 사용됨을 설명한다. 이 때 암호화할 때와는 반대로 key 16은 첫 번째, key 15는 두 번째, ..., key1은 16번째에서 사용됨을 알 수 있다. 자세한 내용은 <2.5. Round함수> 부분에서 확인할 수 있다.
- 이후 다시 암호화의 함수를 그림과 도식으로 이해하기 쉽게 설명하고 있는데, 여기서 한 라운드당 8개의 Sbox가 사용되어 48-bit가 32-bit으로 줄어듦을 확인할 수 있다. 또한 예시로 S1을 통해 어떻게 한 블록당 6-bit가 4-bit가 되는지 자세한 설명을 볼 수 있으며, 그 외에도 bit-selection table, permutation table과 같이 table이 그림으로 나타나있다. 그 외에도 S1~S8, PC-1, PC-2 table들은 뒤쪽 페이지의 부록에 그림으로 첨부되어 있다.
- 앞서 DES의 공격에 대해 잠깐 언급된 후 단순DES의 취약점을 보완하기 위해 만들

어진 3중 DES, Triple Data Encryption Algorithm(TDEA)의 작동원리가 설명되어 있다. TDEA의 암호화 작업은 DES암호화와 복호화로 이루어진 복합 작업이며, TDEA의 암호화작업에서는 우선 64-bit의 블록을 에 따라 블록 O로 변환하고, 복호화(해독)할때는, 64-bit의 블록을 를 따르면 된다. 3개의 키 K1, K2, K3를 사용할 때, 3가지 옵션이 있다. 모두 다른 키인 경우, K1=K3인 경우, K1=K2=K3로 모두 같은 경우가 있는데, 가장 후자의 경우는 우리가 알고 있는 단순 DES라고 할 수 있다. 두 번째 경우에는 단일 DES와 호환이 된다. 이후 앞서 언급된 S-box table과 Permutation table, PC-1, PC-2 table과 이를 통한 키의 압축, 이 과정에서 각 n번째 키가 좌측shift가 1개와 2개중 어떤 값을 가지는지에 대한 설명, TDEA의 두가지 경우에 대한 도식화 그림과 함께 문서가 마무리된다.