제 4 장 대칭 암호알고리즘: DES, SEED128

4.1 DES

4.1.1 DES 개요

대칭 암호알고리즘에 대한 표준의 필요성이 제기됨에 따라 1972년도에 미국의 표준화 기구인 NIST(National Institute of Standards)는 표준화 작업을 착수하였으며, 그 결과로 1977년에 DES(Data Encryption Standard)를 표준으로 공식 채택하였다. DES는 IBM에서 제안한 Lucifer라는 암호알고리즘을 기반으로 하고 있다. NIST는 표준화 과정에서 Lucifer에 대해 NSA(National Security Agency)로부터 자문을 받았으며, 그 결과 원래 키 길이를 128비트에서 56비트로 축소하였고, 내부 동작 메커니즘 중 일부를 교체하였다. 표준으로 채택된 DES의 원래 유효기간은 10년이었지만 20년 동안 표준으로 사용되었다. 그 이유는 10년 후에도 DES가 표준으로 충분한 역할을 할 수 있다고 판단되었기에 기간이 연장된 것이다. 30년이 지난 현재 시점에서도 DES 알고리즘 자체에 대한 큰 허점이나 문제점은 발견되지 않았다. 하지만 56비트 길이의 암호키를 사용하는 DES는 오늘날 컴퓨팅 기술로는 그 키 길이가 너무 짧다. 따라서 1997년에 새 표준에 대한 작업을 시작하였으며 2000년 10월에 새 표준이 채택되었다.

DES는 블록방식의 대칭 암호알고리즘이며, 블록의 크기는 64비트이다. 앞서 언급한 바와 같이 키의 길이는 56비트이다. 하지만 보통 패러티 비트를 추가하여 64비트로 표현한다. 이 키는 총 16개의 길이가 48비트인 부분키로 확장된다. 총 16라운드로 구성되며, 각 라운드는 혼합 암호방식으로 구성되어 있고, 각 라운드마다 하나의 부분키가 사용된다.

4.1.2 DES 암호화/복호화 함수

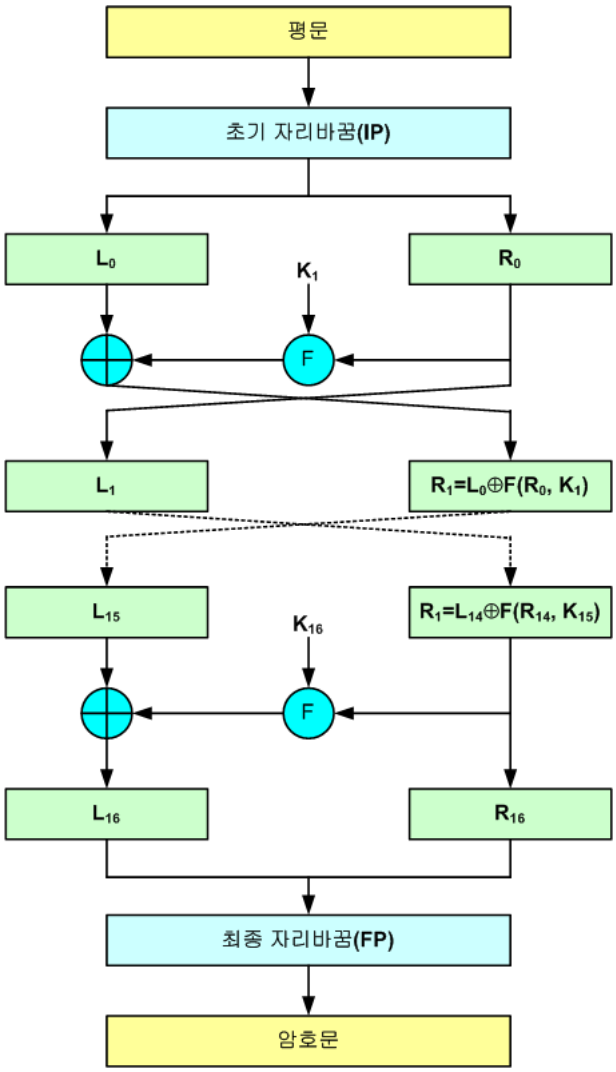
DES의 암호화 과정은 그림 4.1과 같다. 그림에 알 수 있듯이 64비트 평문은 초기 자리바꿈(IP, Initial Permutation) 과정을 거친 후에 두 개의 32비트 블록 L_0, R_0으로 나누어진다. 이들은 16라운드를 거치게 되며, 두 결과 블록은 다시 합쳐 최종 자리바꿈(FP, Final Permutation) 과정을 거쳐 암호문으로 변환된다. 여기서 초기 자리바꿈과 최종 자리바꿈은 서로 역 관계가 성립한다. 즉, 입력을 초기 자리바꿈을 한 다음에 다시 최종 자리바꿈을 하면 그 결과는 입력과 같아진다. 초기 자리바꿈과 최종 자리바꿈은 그림 4.2와 같다. DES는 이미 설명한 바와 같이 총 16라운드를 통해 암호화한다. 이 때 주의할 것은 마지막 라운드는 그 결과가 이전 라운드와 달리 교차되지 않는다. 복호화 과정은 부분키의 사용 순서가 다르다는 것을 제외하고는 암호화 과정과 동일하다. 암호화 과정은 부분키 1부터 16까지 차례로 사용하지만 복호화 과정에서는 반대 순서로 사용한다. DES 과정을 수학식처럼 표현하면 다음과 같다.

$$DES(M, K) = \mathrm{IP}^{-1} J_{16}^{K_{16}} \ldots J_2^{K_2} J_1^{K_1} \mathrm{IP}(M)$$

$$DES^{-1}(C, K) = \mathrm{IP}^{-1} J_{16}^{K_1} \ldots J_2^{K_{15}} J_1^{K_{16}} \mathrm{IP}(C)$$

$$J_i^{K_i}(L_{i-1}\|R_{i-1}) = L_i\|R_i$$

여기서 $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, $\mathrm{IP}(M) = L_0 R_0, C = \mathrm{IP}^{-1}(R_{16} L_{16})$이다.



<그림 4.1> DES의 암호화 과정

초기 자리바꿈과 최종 자리바꿈은 서로 상쇄되는 연산이므로 암호학적으로 아무런 의미가 없다. 따라서 DES의 안전성은 각 라운드에서 사용되는 F 함수에 의존한다. F 함수의 내부 구성은 그림 4.3과 같다. F 함수는 입력으로 받은 32비트 블록을 확장 자리바꿈을 통해 48비트로 확장한 다음 부분키와 XOR 연산을 한다. 그 다음 결과를 S-box 치환을 통해 32비트로 축소하고, 이 결과를 다시 P-box 자리바꿈을 한다. DES에서 사용되는 대부분의 연산은 역이 가능하다. 유일하게 역을 취할 수 없는 연산이 S-box 치환이다. 따라서 DES 안전성에 매우 중요한 역할을 하는 것이 S-box이다. 확장 자리바꿈과 P-box 자리바꿈은 그림 4.4와 같다. 확장 자리바꿈의 경우에는 새로운 값을 이용하여 32비트를 48비트로 확장하는 것이 아니라 기존 비트들 중에 일부를 중복하여 확장을 하고 있다. 이와 같이 확장하는 이유는 내부 연산의 적용이 빠르게 퍼지도록 하기 위함이다. 이와 같은 효과를 눈사태 효과

(avalanche effect)라 한다. 따라서 매우 유사한 평문을 암호화하여도 그 결과는 매우 다르다.

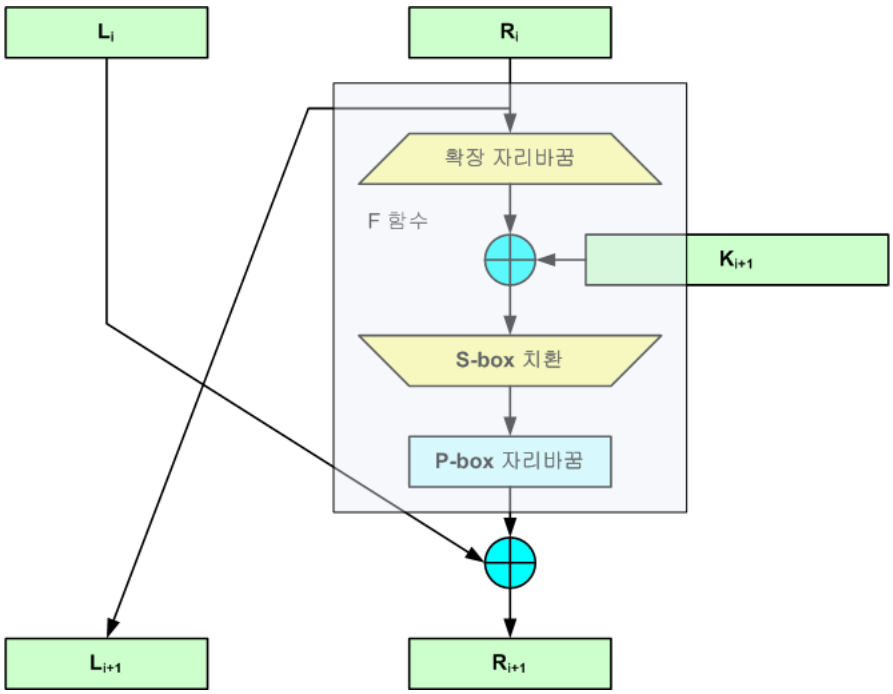
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(ㄱ) 초기 자리바꿈

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

(ㄴ) 최종 자리바꿈

<그림 4.2> 초기 자리바꿈과 최종 자리바꿈



<그림 4.3> DES의 F 함수의 구성도

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

(ㄱ) 확장 자리바꿈

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

(ㄴ) P-Box 자리바꿈

<그림 4.4> 확장 자리바꿈과 P-Box 자리바꿈

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

<그림 4.5> S-box 1

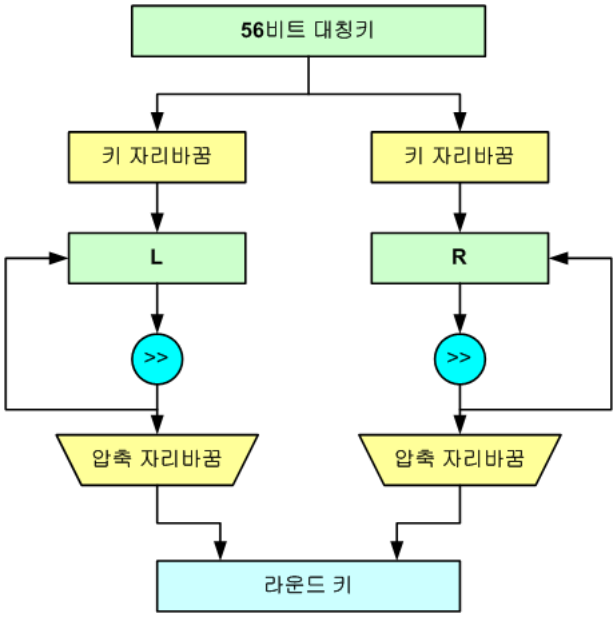
DES에는 총 8개의 S-Box가 정의되어 있다. 각 S-box 치환은 6비트를 입력받아 4비트를 출력한다. 첫 번째 S-box은 그림 4.5와 같다. 그림에서 알 수 있듯이 각 S-box의 모든 행은 0부터 15까지 수로 구성되어 있다. S-box의 치환은 다음과 같은 방법으로 이루어진다.

- 입력 6비트를 $b_6b_5b_4b_3b_2b_1$로 표현하였을 때 이중에 b_6b_1이 행 값이 되고, $b_5b_4b_3b_2$가 열 값이 된다. 예를 들어 입력이 011011이면 행은 1(01)이고, 열은 13(1101)이 된다. 따라서 결과는 5(0101)가 된다.

이와 같은 방법을 사용하므로 출력 값이 5이면 그것의 입력은 011000, 011011, 111100, 110001 중 하나가 된다. 따라서 역을 취할 수 없다.

4.1.3 DES 키 스케줄링

DES는 이미 설명한 바와 같이 56비트를 사용한다. 하지만 8개의 패리티 비트를 추가하여 보통 64비트로 표현한다. DES의 56비트 키는 16개의 부분키로 확장되어 각 라운드에 사용된다. 이 처럼 하나의 짧은 길이의 키로부터 각 라운드에 사용할 키를 생성하는 방법을 키 스케줄링 알고리즘이라 한다. DES의 키 스케줄링 알고리즘은 그림 4.6에 기술되어 있으며, 다음과 같다.



<그림 4.6> DES의 키 스케줄링 알고리즘

- **단계 1.** 56비트 키를 그림 4.7에 기술된 키 자리바꿈 연산을 통해 두 개의 28비트 블록으로 나눈다.
- **단계 2.** 두 개의 블록은 각 라운드마다 독립적으로 1비트 또는 2비트 왼쪽 순환이동을 한다. 정확하게 말하면 1, 2, 9, 16번째 라운드 키를 생성할 때를 제외하고는 모두 2비트 이동을 한다.
- **단계 3.** 각 28비트 블록은 그림 4.8에 기술된 압축 자리바꿈을 통해 24비트 블록이 되며, 두 개의 24비트 블록이 결합되어 48비트 라운드 키가 생성된다.
- **단계 4.** 단계 2부터 3을 16번 수행하여 각 라운드 키를 생성한다.

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

(ㄱ) 왼쪽 키 자리바꿈 맵

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

(ㄴ) 오른쪽 키 자리바꿈 맵

<그림 4.7> 키 자리바꿈

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

(ㄱ) 왼쪽 압축 자리바꿈 맵

41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

(ㄴ) 오른쪽 압축 자리바꿈 맵

<그림 4.8> 키 자리바꿈

4.1.4 DES의 안전성

DES 내부에서 사용되는 연산은 크게 XOR, 자리바꿈, 순환이동, 치환 등이 있다. 이 중에 유일하게 선형이 아닌 것은 S-Box 치환 연산이다. 즉, 나머지 연산들은 그 역을 취할 수 있을 뿐만 아니라 기지 평문 공격을 통해 키를 알아낼 수 있다. 따라서 S-Box가 DES의 안전성에 가장 중요한 부분이다. NSA가 표준에 대한 자문 과정에서 원래 Lucifer가 사용하던 것을 현재의 S-Box로 바꾸었다. 이와 같은 이유 때문에 NSA가 이 부분에 자신들만이 알고 있는 트랩도어를 숨겼을 것이라는 소문이 있으며, 거꾸로 NSA가 IBM을 믿지 못하였기 때문에 변경하였을 것이라는 소문도 있다. 대칭 암호알고리즘에 대한 가장 강력한 해독 공격 중 하나가 차분 해독법(differential cryptoanalysis)이다. 그런데 Biham과 Shamir는 DES의 S-Box가 차분 해독법에 대해 강건하도록 설계되어 있음을 나중에 증명하였다. 이 증명이 공개되자 IBM 개발자는 이미 개발할 당시부터 이 해독공격 방법을 알고 있었지만 공표하지 않았다고 주장하였다.

이 처럼 표준으로 채택 된지 30년이 지난 오늘날까지 DES에 대한 어떤 허점도 알려지지 않고 있다. 하지만 DES의 키 길이는 56비트밖에 되지 않으며, 오늘날 컴퓨팅 능력은 DES가 표준으로 채택된 1977년과는 비교되지 않는다. 1977년 당시 Diffie와 Hellman은 $20,000,000이면 하루만에 DES 키를 찾아내는 특수 하드웨어를 제작할 수 있음을 보였다. 비용에서 알 수 있듯이 현실적인 보안 문제로 보기 어렵다. 하지만 1993년에 Weiner는 $100,000이면 하루 반 만에 DES 키를 찾아내는 특수 하드웨어를 제작할 수 있음을 보였다. 따라서 DES 알고리즘을 그대로 활용하면서 DES의 키 길이를 확장하는 방안이 필요하게 되었다. 이에 제안된 것이 **3중 DES**(Triple DES)이다.

앞서 언급한 바와 같이 DES에는 큰 허점이 없지만 DES에서 사용 가능한 2^{56}개의 키 중에 일부를 사용할 경우에는 안전성을 보장할 수 없다. DES는 16개의 라운드 키를 사용하는데 이들이 모두 다를 경우가 가장 바람직하다. 반대로 이들이 모두 같은 경우가 발생할 수 있는데 이 경우에는 안전성에 치명적인 문제를 야기할 수 있다. 표 4.1에 제시된 4개의 키를 사용하면 라운드 키가 모두 같아지는 경우가 발생한다.

<표 4.1> DES의 약한 키

약한 키(홀수 패리티 사용)	실제 키 (키 자리바꿈 후)
01 01 01 01 01 01 01 01	00 00 00 00 00 00 00
FE FE FE FE FE FE FE FE	FF FF FF FF FF FF FF
1F 1F 1F 1F 0E 0E 0E 0E	00 00 00 0F FF FF FF
E0 E0 E0 E0 F1 F1 F1 F1	FF FF FF F0 00 00 00

이들을 키를 사용할 경우에 평문을 암호화한 결과가 평문 자체인 경우가 발생할 수 있다. 이들을 **약한 키**(weak key)라 한다. 이 외에 12개의 **중간 약한 키**(semiweak key)들이 있다. 이들을 사용하면 두 종류의 라운드 키만 사용하게 되며, 이들 키를 사용할 경우에 평문을 암호화한 결과가 평문 자체의 보수가 될 수 있다. 또 4개의 라운드 키들만 사용하게 되는 48개의 키가 존재한다. 따라서 DES 키를 생성할 때 임의로 생성된 키가 이들에 속하는지 검사할 수 있다. 하지만 여기서 주목해야 할 것은 가능한 2^{56}개의 키 중에 64개만 문제가 되므로 이것이 큰 문제가 된다고 할 수는 없다.

DES의 보안성을 위협하는 또 한 가지 특성은 보수 특성(complementation property)이다. 보수 특성이란 DES에서 다음이 성립한다는 것을 말한다.

$$E_K(P) = C,\ E_{\overline{K}}(\overline{P}) = \overline{C}$$

즉, 어떤 평문 P를 키 K로 암호화하였을 때 결과가 C라 할 때, 평문을 보수하고 키를 보수하여 암호화하면 그 결과는 C의 보수가 된다. 이것의 의미는 2^{56}개의 키 대신에 2^{55}개의 키만 검사하면 키를 찾을 수 있다는 것을 의미한다. 그 이유는 다음과 같이 설명할 수 있다.

- 어떤 사용자가 키 K를 사용한다고 가정하고, 이 사용자가 이 키로 암호화한 평문과 암호문 쌍 $(P, C_1), (\overline{P}, C_2)$ 두 개를 공격자가 가지고 있다고 가정하자.
- 이 경우 공격자는 하나의 키 K'을 검사하면 K'과 $\overline{K'}$가 사용자의 키인지 동시에 다음과 같이 확인할 수 있다.
  - 우선 추측한 키 K'를 이용하여 P를 암호화한다. 그 결과가 $C = E_{K'}(P)$라 하자.
  - 이 때 $C = C_1$이면 $K' = K$가 된다.
  - 만약 $\overline{C} = C_2$이면 다음과 같은 이유 때문에 $\overline{K'} = K$가 된다. $C = E_{K'}(P)$이므로 보수 특성에 의해 $E_{\overline{K'}}(\overline{P}) = \overline{C}$이다. 그런데 $\overline{C} = C_2$이므로 $E_{\overline{K'}}(\overline{P}) = C_2$이다.

이 처럼 DES의 보수 특성은 DES의 안전성에 영향을 준다. 하지만 DES는 군이 아니다. 만약 DES가 군 특성을 가지고 있으면 삼중/이중 DES는 그 효과를 기대하기 어렵다. DES는 군과 관련하여 다음 두 가지 특성을 가지고 있다.

- $\neg \exists\, K_3 \,\mathrm{s.t}\ E_{K_2}(E_{K_1}(P)) = E_{K_3}(P)$
- $\neg \exists\, K_4 \,\mathrm{s.t}\ E_{K_3}(E_{K_2}(E_{K_1}(P))) = E_{K_4}(P)$

4.2 블록 암호알고리즘의 구조

블록 암호방식은 앞서 살펴본 DES처럼 단순 연산들을 구성된 단순한 함수를 반복적으로 적용하여 암호학적으로 강한 함수를 만드는 과정으로 개발된다. 반복되는 함수를 라운드 함수라 하고, 이 함수에서 사용되는 암호키를 라운드 키라 한다. 라운드 키는 서로 독립적인 것이 바람직하지만 키 길이가 너무 길어져 보통 하나의 짧은 암호키를 확장하여 라운드 키를 생성한다. 블록 암호방식은 라운드 함수를 반복적으로 적용하는 방법에 따라 크게 SPN(Substitution-Permutation Network)과 Feistel 구조로 분류된다. DES는 Feistel 구조에 속한다. 특히, DES의 복호화 과정에서 암호문이 평문으로 바뀌는 과정을 면밀히 관찰하면 복호화 과정에서 암호화 과정에 계산된 F함수의 값들과 동일한 값들을 만들어 XOR 연산을 통해 상쇄시키고 있다는 것을 알 수 있다. 즉, F함수가 어떤 모습을 하던지 Feistel 구조는 역 변환이 가능하다.

4.3 대칭 암호알고리즘에 대한 공격

대칭 암호알고리즘에 대해 지금까지 알려진 가장 강력한 공격 방법들은 차분 해독법과 선형 해독법(linear cryptoanalysis)이다. 차분 해독법은 두 개의 평문의 XOR 값과 대응되는 두 개의 암호문의 XOR 값을 비교하여 공격하는 방법이며, 선형 해독법은 평문에 있는 비트들의 선형 관계와 마지막 S-box의 입력 비트들의 선형 관계를 통해 공격하는 방법이다.

4.4 DES의 확장

앞서 언급한 바와 같이 DES의 가장 큰 문제는 키 길이가 짧아 오늘날 컴퓨팅 능력으로는 쉽게 전사공격을 할 수 있다는 것이다. 이를 극복하기 위해 DES를 확장하는 방법에 대해 연구되었다. 이 절에서 DES의 확장인 이중 DES와 삼중 DES를 살펴본다.

4.4.1 이중 DES

이중 DES는 하나의 평문을 암호화할 때 다음과 같이 서로 다른 두 개의 키로 두 번 암호화하는 것이다.

- 암호화: $C = E_{K_2}(E_{K_1}(P))$
- 복호화: $P = D_{K_1}(D_{K_2}(C))$

앞서 언급한 바와 같이 DES는 군 성질을 가지고 있지 않으므로 이 처럼 암호화하면 키의 길이를 56비트에서 112비트로 확장이 가능하다고 생각할 수 있다. 하지만 다음과 같은 **중간만남 공격**(meet-in-the-middle attack) 때문에 안전성이 단일 DES와 별 차이가 없다.

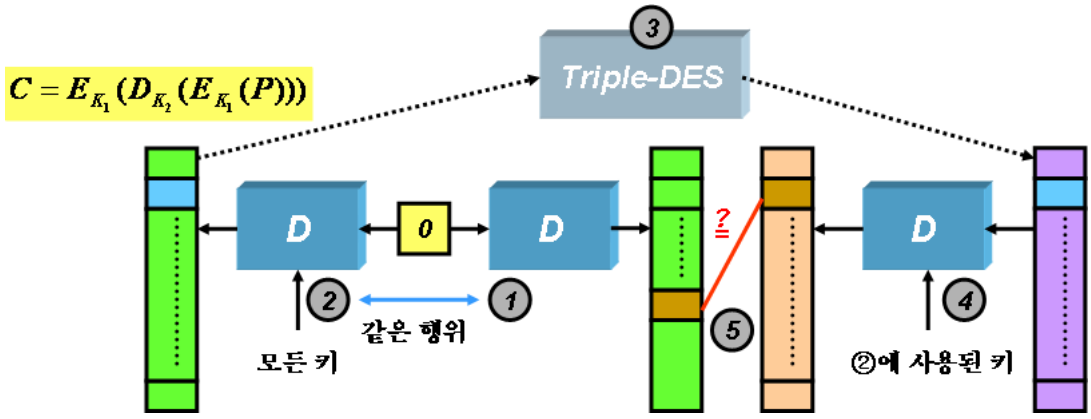
이중 DES에 대한 중간 만남 공격은 $E_{K_1}(P) = D_{K_2}(C)$와 같다는 특성을 이용한 공격이다. 즉, 기지 평문 쌍 (P, C)를 가지고 있는 공격자는 P를 우선 가능한 모든 키로 암호화하여 목록을 만들고, C를 가능한 모든 키로 복호화하여 또 다른 목록을 만든다. 두 목록을 비교하여 일치하는 쌍을 찾으면 그 때 사용된 두 개의 키가 후보 키가 될 수 있다. 공격자가 하나의 기지 평문 쌍만 가지고 있을 경우에는 248개는 엉뚱한 후보를 만날 수 있다. 이것은 한 암호문을 생성할 때 사용될 가능한 키의 개수가 2112이지만 가능한 암호문의 수는 264이므로 248개의 키는 한 평문을 동일한 암호문으로 암호화될 수 있다. 하지만 공격자가 기지 평문 쌍을 두 개만 가지고 있으면 가능한 암호문의 수가 2128로 증가하므로 서로 다른 두 개의 키를 이용하여 평문을 암호화하였을 때 결과 암호문이 같을 확률은 2-16이다. 따라서 두 개의 기지 평문 쌍만 있으면 258의 노력으로 정확하게 찾을 수 있다.

4.4.2 삼중 DES

앞 절에 살펴본 바와 같이 이중 DES는 중간 만남 공격 때문에 기대했던 $O(2^{112})$의 안전성 대신에 단일 DES와 별 차이가 없는 $O(2^{58})$의 안전성만 제공한다. 따라서 이를 극복하기 위해 삼중 DES가 제안되었다. 삼중 DES는 DES를 세 번 사용하지만 세 개의 다른 키 대신에 두 개의 키만 다음과 같이 사용한다.

- 암호화: $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$
- 복호화: $P = D_{K_1}(E_{K_2}(D_{K_1}(C)))$

위 식을 알 수 있듯이 세 번 연속해서 암호화를 하지 않고 중간에 복호화 연산을 사용한다. 이것은 두 개의 키가 같을 경우에는 단일 DES와 같아지기 때문에 호환성을 위해 이와 같이 정의되었다. 이와 같은 형태로 암호화하면 이중 DES와 달리 중간 만남 공격이 가능하지 않다.



<그림 4.9> Merkle과 Hellman의 삼중 DES에 대한 선택 평문 공격

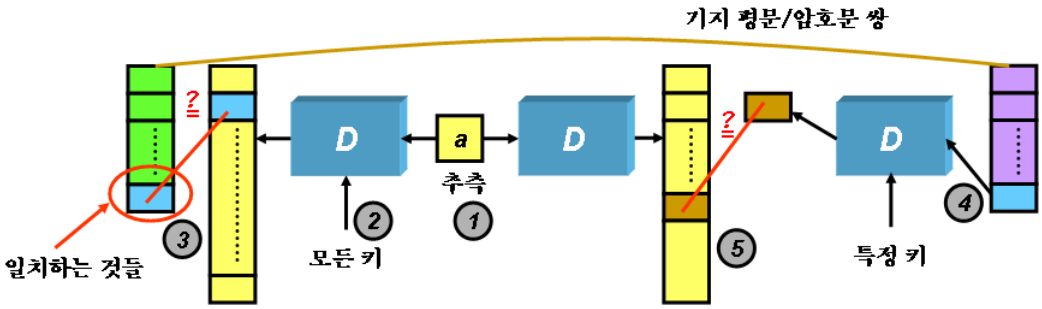
Merkle과 Hellman은 복잡도가 O(2^{57})인 삼중 DES에 대한 선택 평문 공격을 발견하였다. 그러나 이 공격은 2^{56}개의 선택 평문/암호문 쌍이 필요하므로 현실성이 있는 공격이 아니다. 이 공격의 원리는 그림 4.9에 기술되어 있으며, 다음과 같다.

- **단계 1.** 모든 비트가 0인 64비트 블록을 만들어 이것을 K_1으로 암호화된 암호문이라고 가정한다. 여기서 이 블록의 비트 값은 꼭 모두 0일 필요는 없다. 이 블록을 가능한 모든 키로 복호화하여 평문 목록을 만든다, 이 목록은 평문을 K_1으로 암호화하고 K_2로 복호화한 목록으로도 활용한다.
- **단계 2.** 공격자는 만들어진 평문 목록에 대한 선택 평문 공격을 할 수 있다고 가정한다. 즉, 이 평문 목록에 대응되는 삼중 DES의 암호문 목록을 공격자가 얻을 수 있다고 가정한다.
- **단계 3.** 삼중 DES의 선택 평문 암호문 목록을 대응되는 평문을 얻을 때 사용된 키로 복호화한다.
- **단계 4.** 단계 3에서 복호화하여 얻은 목록과 평문 목록을 비교하여 일치된 값을 찾는다.

단계 1의 비용은 2^{56}이며, 단계 3의 비용도 역시 2^{56}이다. 따라서 총 비용은 2^{57}이다. 이 공격은 앞서 언급한 바와 같이 2^{56}개의 선택 평문이 필요하므로 현실성이 없다.

Oorshot와 Wiener는 선택 평문 공격 대신에 기지 평문 공격이 가능하다는 것을 보였다. Merkle과 Hellman은 삼중 DES 암호화 과정의 중간 값을 고정한 상태에서 출발하므로 선택

평문 공격을 해야 하였다. 따라서 Oorshot와 Wiener는 이것을 기지 평문 공격으로 바꾸기 위해 중간 값을 고정하지 않는다. 대신 추측을 한 다음에 가능한 모든 키로 복호화하여 가지고 있는 기지 평문과 비교하여 일치하는 것을 찾으면 Merkle과 Hellman 방법과 동일한 방법으로 공격할 수 있다. 그러면 추측하여 성공할 확률이 어떻게 되는지 살펴보자. 하나의 기지 평문 쌍이 있을 때 올바른 중간 값을 선택할 확률은 $1/2^{64}$이다. 따라서 n개의 쌍이 있으면 성공할 확률은 $n/2^{64}$이다. 그러므로 성공하기 위해서는 대략 $2^{64}/n$번 시도를 해야 한다. 따라서 이 공격의 비용은 대략 $O(2^{56} \times 2^{64}/n)$이며, 기지 평문 쌍이 많을수록 비용이 줄어든다. 하지만 여전히 많은 기지 평문 쌍이 필요하므로 삼중 DES의 사용이 문제가 될 정도의 현실성이 있는 공격은 아니다.



<그림 4.10> Oorshot와 Wiener의 삼중 DES에 대한 기지 평문 공격

삼중 DES에 암호화 모드 적용하는 방법은 크게 두 종류가 있다. 하나는 삼중 DES를 하나의 암호화 함수로 보고 암호화 모드를 적용하는 것이고, 다른 하나는 세 번의 DES 암호화마다 암호화 모드를 적용하는 것이다. 전자를 외부 암호화 모드 방법이라 하고, 후자를 내부 암호화 모드 방법이라 한다. 외부 방법은 하나의 초기벡터만 필요하지만 내부 방법은 3개의 초기벡터가 필요하다. 따라서 결과 암호문이 3블록이나 커진다. 만약 하드웨어로 삼중 DES를 구현하고, 3개의 DES 칩을 사용한다면 내부 방법은 각 칩을 병행 수행하여 빠르게 암호화할 수 있지만 외부 방법은 순차적으로 사용할 수밖에 없다. 하지만 안전성은 오히려 외부 방법이 더 안전하다고 알려져 있다.

참고문헌

[1] Data Encryption Standard (DES), Federal Information Processing Standards Publication 46, Reaffirmed, Oct. 1999.

[2] W. Tuchmann, "Hellman Presents No Shortcut Solutions to DES," IEEE Spectrum, Vol. 16, No. 7, pp. 40-41, 1979.

[3] R.C. Merkle and M. Hellman, "On the Security of Multiple Encryption," Communications of the ACM, Vol. 24, No. 7, pp. 465-467, 1981.

[4] P.C. van Oorshot and M.J. Wiener, "A Known-Plaintext Attack on Two-key Triple Encryption," Advances in Cryptology, Eurocrypt 1990, LNCS Vol. 473, pp. 318-325, 1991.

연습문제

1. DES의 보수 특성 $E_K(M) = \overline{E_{\overline{K}}(\overline{M})}$을 증명하시오.
2. DES의 암호화와 복호화가 올바르게 동작한다는 것을 증명하시오. 즉, $D_K(E_K(M)) = M$ 이 성립함을 증명하시오.