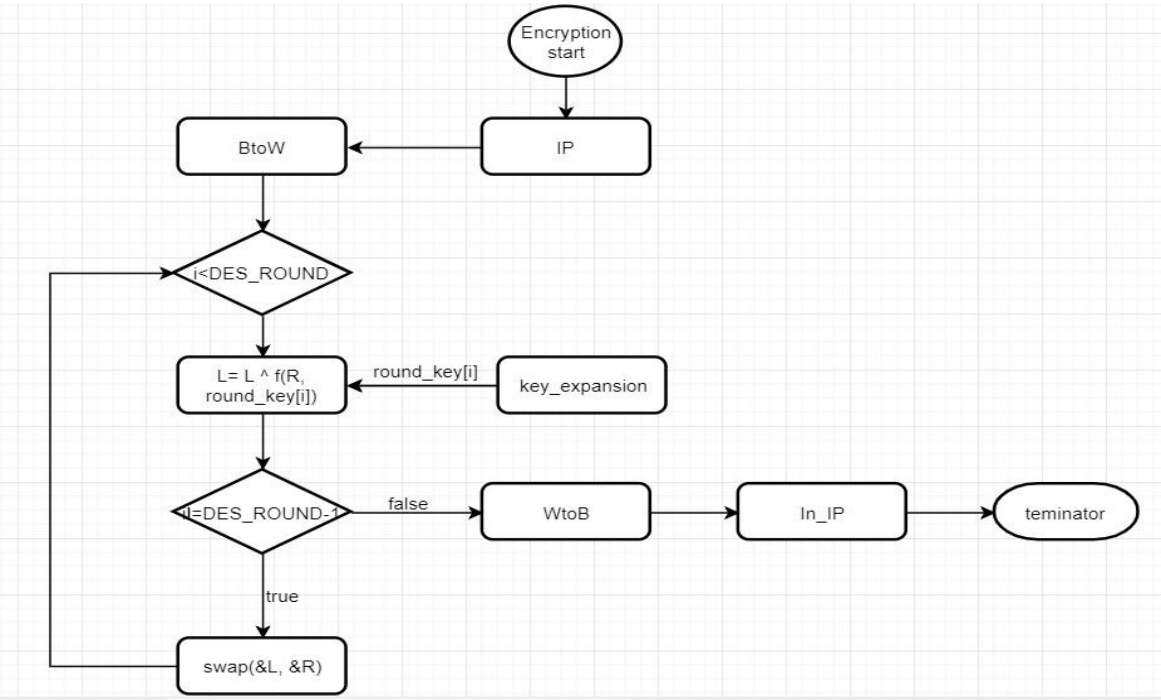
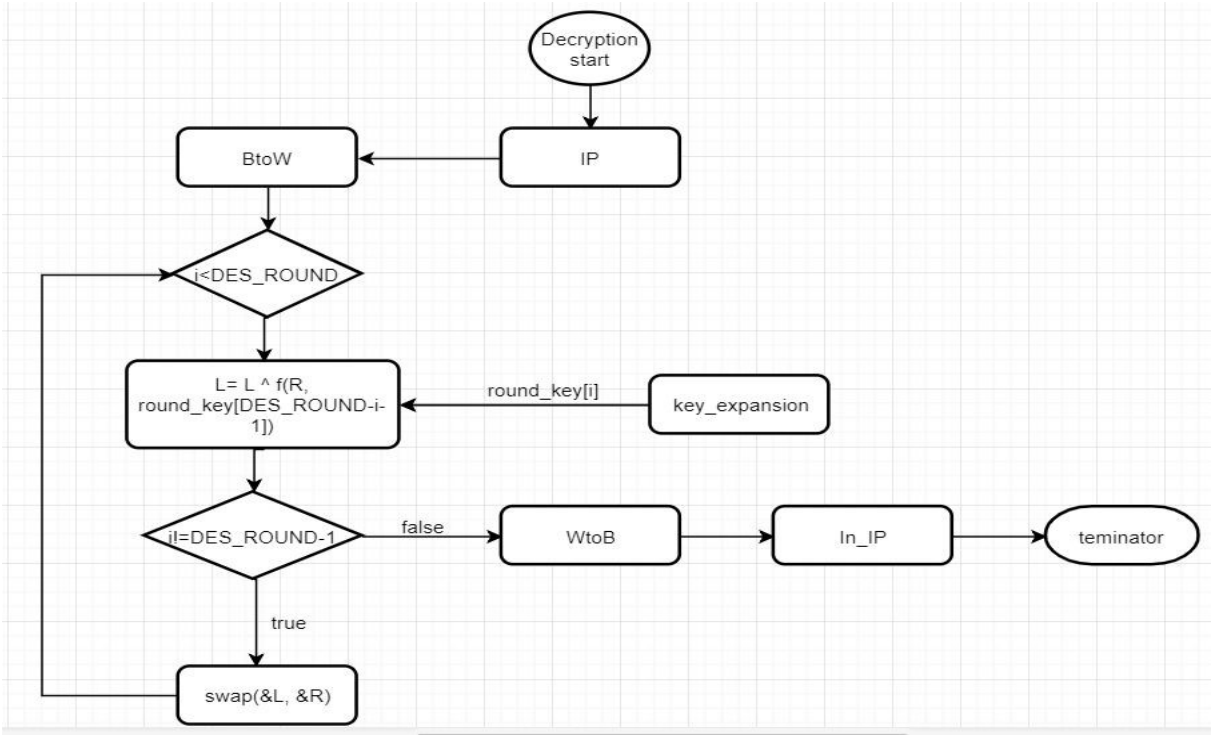
1. 직접 구현한 DES 암호 알고리즘에 나오는 각 함수들 간의 플로우 차트를 그려보시오.

DES Encryption :



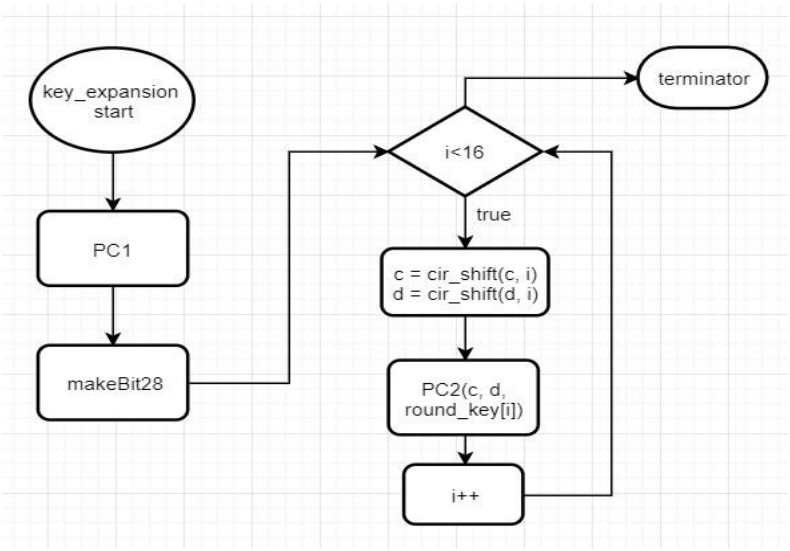
DES는 64비트(8바이트)의 블록을 입력 받아 암호문을 만들어 낸다. DES의 암, 복호화의 처음과 마지막은 초기치환과 최종치환을 수행하는데 두 치환은 서로 역의 관계이다. 우선 처음에 64비트 블록이 입력되면 IP라는 초기치환을 거친다. 초기치환은 64비트를 입력 받아 미리 정의된 규칙에 재배열한다. DES의 마지막은 In_IP는 라는 최종치환을 통해 최종 암호문을 만들어 낸다. DES에서 초기치환과 최종치환은 암호학적으로 의미가 없다. IP(초기치환) 과정이 끝나면 BtoW라는 함수가 실행된다. 해당 함수는 64비트 블록을 32비트 2개의 블록으로 나눠주는 단순한 함수이다. 해당 함수의 결과로 L, R을 구할 수 있고 라운드 함수의 입력 값으로 사용된다. DES는 16번의 라운드 함수를 사용한다. DES의 각 라운드 함수는 Feistel 암호 구조로 되어있고, 라운드 함수의 f는 32비트 출력 값을 산출하기 위하여 가장 오른쪽의 32비트에 48비트 키를 적용한다. 라운드 함수의 f는 확장 P-박스, 키 XOR, 8개의 S-박스, 단순 P-박스 4가지로 구성된다. 가장 오른쪽의 32비트인 R이 인자로 전달되면 확장 P-박스를 통해 32비트를 48비트로 확장한다. 그런 다음 round key generator을 통해서 생성된 round key와 키 XOR 연산을 하고, 8개의 S-박스의 입력 값으로 전달된다. 8개의 S-박스는 실제로 섞어주는 역할을 수행함으로 confusion 역할을 수행한다. 해당 함수가 완료되면 48비트 데이터가 32비트로 축소된다. 마지막으로 32비트 데이터를 단순 P-박스의 입력 값으로 전달하여 최종적으로 치환을 마치면 라운드 함수의 f가 동작을 마친다. 이와 같은 동작을 16번 반복하고, 매 라운드 함수 마지막에 swapper를 호출해서 L과 R을 swap해준다. 마지막 라운드에서는 swap을 하지 않는다. DES의 라운드 함수가 끝나면 WtoB가 호출되고, 2개의 32비트 데이터를 64비트 데이터로 만든다. 그런 다음 In_IP(최종치환)을 통해서 최종적인 암호문을 생성한다.

DES Decryption :



DES의 decryption은 encryption과 구조가 동일한다. 단지 복호화 과정에서 라운드 키들이 역순으로 적용되어야 한다. 즉 암호화 과정에서 1라운드에 K1을 사용하고 16라운드는 K16을 사용했다면, 복호화 과정에서는 1라운드에 K16을 사용하고 16라운드는 K1을 사용해야 된다. 라운드 키를 역순으로 사용하기 위해 라운드 함수의 f 함수의 인자 값으로 라운드 키를 역순으로 넣어준다. 암호화 과정처럼 16번의 라운드 함수가 끝나면 최종적인 plain text를 얻을 수 있다.

Round Key Generator :



DES함수는 보통 64비트 키가 주어지고, 64비트 중 8비트는 패리티 비트로 실제 키 생성 과정

전에 패리티 제거 치환인 축소 P-박스를 통해 제거된다. 본 프로그램에서는 PC1함수를 통해 수행된다. PC1함수의 반환 값으로 56비트 키가 주어지면 makeBit28 함수를 호출해서 56비트 키를 2개의 28비트 키로 만든다. 각 28비트 키는 매 라운드 마다 비트를 왼쪽으로 좌측 순환 이동한다. 1,2,9,16번째 라운드에서는 1비트만 좌측으로 순환 이동하고, 나머지 라운드에서는 2비트를 좌측으로 순환 이동한다. 이러한 과정을 DES의 라운드 함수가 16번 실행되는 것처럼 16번 반복한다. 좌측 28비트와 우측 28비트가 cir_shift함수를 통해 좌측 순환이동이 끝나면 28비트 키 2개를 합쳐서 DES의 라운드 함수 라운드 키로 사용하기 위해 48비트 키를 만들어 낸다. 48비트 키를 만들어 내기 위해 축소 치환이 사용되고, 본 프로그램에서는 PC2 함수가 축소 치환을 수행한다. PC2의 반환 값이 DES의 한 라운드의 라운드 키로 사용된다.

2. 직접 구현한 DES 암호를 사용하여 교재 Chapter 6의 [예제 6.5]에 나오는 평문과 키를 사용하여 암호화 <Table 6.15>를 그대로 재현해 보고 그 결과를 상세히 분석하시오

```
*평문 입력 : 123456ABCD132536
*비밀키 입력 : AABB09182736CCDD
DES Encryptioning...
Round 1 = 18ca18ad : 671e37b8
Round 2 = 671e37b8 : 2a7a1ffa
Round 3 = 2a7a1ffa : 9e4c9437
Round 4 = 9e4c9437 : c6ce11dd
Round 5 = c6ce11dd : 9f61ab15
Round 6 = 9f61ab15 : 8a545383
Round 7 = 8a545383 : c2613fda
Round 8 = c2613fda : 3659ee4d
Round 9 = 3659ee4d : e2fc4fc2
Round 10 = e2fc4fc2 : aa5df67d
Round 11 = aa5df67d : 33b95169
Round 12 = 33b95169 : 17f9ba70
Round 13 = 17f9ba70 : 50aac840
Round 14 = 50aac840 : 392074c3
Round 15 = 392074c3 : c2f7fae0
Round 16 = 1930d911 : c2f7fae0
after combination : 1930d911c2f7fae0
* 암호문 : 65a8204c7d3aaeae
DES Decryptioning...
Round 16 = c2f7fae0 : 392074c3
Round 15 = 392074c3 : 50aac840
Round 14 = 50aac840 : 17f9ba70
Round 13 = 17f9ba70 : 33b95169
Round 12 = 33b95169 : aa5df67d
Round 11 = aa5df67d : e2fc4fc2
Round 10 = e2fc4fc2 : 3659ee4d
Round 9 = 3659ee4d : c2613fda
Round 8 = c2613fda : 8a545383
Round 7 = 8a545383 : 9f61ab15
Round 6 = 9f61ab15 : c6ce11dd
Round 5 = c6ce11dd : 9e4c9437
Round 4 = 9e4c9437 : 2a7a1ffa
Round 3 = 2a7a1ffa : 671e37b8
Round 2 = 671e37b8 : 18ca18ad
Round 1 = 14a7d678 : 18ca18ad

* 복호문 : 123456abcd132536
```

평문이 주어지면 초기치환을 통해서 평문과 다른 64비트 데이터로 변환한다. 변환된 데이터

를 L과 R로 나눈 후에 DES의 라운드 함수의 입력 값으로 전달한다. DES의 라운드 함수는 f함수의 입력 값으로 R을 넣어주면 f 함수의 subroutine인 확장 P-박스, 키 XOR, 8개의 S-박스, 단순 P-박스가 순차적으로 실행이 되고 R1이 생성된다. 좌측의 32비트인 L1은 단순하게 f 함수의 결과 값인 R1과 XOR한 후 swapper를 통해서 위치를 바꾼다. 그러면 Round 2에서는 L1이 Round 2의 라운드 함수의 R2가 되고, R1이 Round 2의 라운드 함수의 L2가 된다. 이러한 내용을 위에 프로그램 실행 결과로 확인할 수 있다. 암호화의 Round 16은 L과 R을 swapping을 하지 않는다. 위에 프로그램의 Round 15와 Round 16을 보면 Round 15의 결과 값(Round 15는 swapping을 한다)이 Round 16의 입력 값으로 전달되고, Round 16에서는 R15의 결과 값이 R16이 되는 것을 확인할 수 있다. 이와 같은 16번의 DES라운드 함수를 끝마치고 2개의 32비트 데이터를 64비트로 합친 것이 after combination의 결과 값이다. 그런 다음 64비트 데이터를 최종치환으로 전달하고, 최종치환을 통해서 최종적인 암호문을 생성한다. DES의 복호화는 암호화와 구조가 동일하고 단지 라운드 키의 적용 순서가 역순이기 때문에 라운드 키의 역순 적용을 보여주기 위해 출력 결과를 Round 16부터 보여주고 있다.

3. 직접 구현한 DES 암호를 사용하여 교재 교재 Chapter 6의 [예제6.6]에 나오는 암호문과 키를 사용하여 복호화<Table 6.16>를 그대로 재현해 보고 그 결과를 상세히 분석하시오.

```
1. 평문, 키 입력 시 - 암호화 - 복호화 과정 출력 2. 암호문, 키 입력 시 복호화 과정 출력 : 2
*암호문 입력 : 65a8204c7d3aaeae
*비밀키 입력 : AABB09182736CCDD
DES Decryptioning...
Round 16 = c2f7fae0 : 392074c3
Round 15 = 392074c3 : 50aac840
Round 14 = 50aac840 : 17f9ba70
Round 13 = 17f9ba70 : 33b95169
Round 12 = 33b95169 : aa5df67d
Round 11 = aa5df67d : e2fc4fc2
Round 10 = e2fc4fc2 : 3659ee4d
Round 9 = 3659ee4d : c2613fda
Round 8 = c2613fda : 8a545383
Round 7 = 8a545383 : 9f61ab15
Round 6 = 9f61ab15 : c6ce11dd
Round 5 = c6ce11dd : 9e4c9437
Round 4 = 9e4c9437 : 2a7a1ffa
Round 3 = 2a7a1ffa : 671e37b8
Round 2 = 671e37b8 : 18ca18ad
Round 1 = 14a7d678 : 18ca18ad

* 복호문 : 123456abcd132536
계속하려면 아무 키나 누르십시오 . . .
```

복호화 과정에서는 암호화 과정에서 사용된 라운드 키가 역순으로 DES의 라운드 함수에 적용된다. 복호화 하는 동안 L0과 R0의 값은 암호화하는 동안에 생성된 L16과 R16과 같다. 이것은 다른 라운드들에 대해서도 동일하게 적용되는 것을 위에 실행결과를 통해서 확인할 수 있다. 즉 DES의 암호화의 각 라운드와 대응되는 복호화의 각 라운드는 역의 관계임을 프로그래밍 실행의 결과로 확인할 수 있다. 또한 초기치환 단계와 최종치환도 역시 암호화와 복호화 과정에서 생성된 값에 역의 관계임을 실행 결과로 확인할 수 있다.

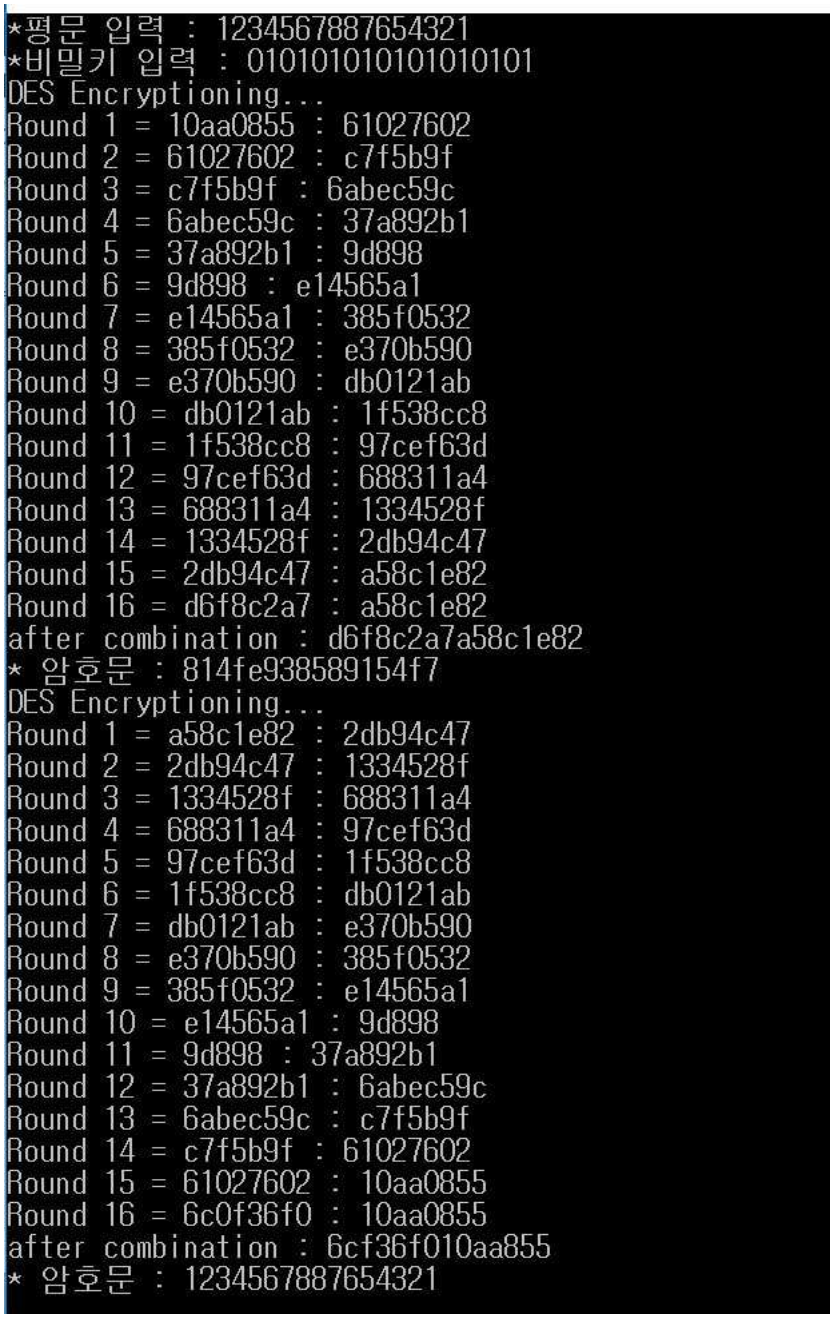
4. 직접 구현한 DES 암호를 사용하여 교재 Chapter 6의 [예제6.7]를 그대로 재현해 보고 그 결과를 상세히 분석하시오.

```
*평문 입력 : 1111111111111110
*비밀키 입력 : 22234512987ABB23
DES Encryptioning...
Round 1 = 0 : d827dbc3
Round 2 = d827dbc3 : 8c646118
Round 3 = 8c646118 : f7acf476
Round 4 = f7acf476 : 6755e21f
Round 5 = 6755e21f : 6b8df762
Round 6 = 6b8df762 : 2ca846dd
Round 7 = 2ca846dd : 9d1adbf3
Round 8 = 9d1adbf3 : cd0a39f3
Round 9 = cd0a39f3 : 88bcd59b
Round 10 = 88bcd59b : cf1388ad
Round 11 = cf1388ad : c243c737
Round 12 = c243c737 : 550e920a
Round 13 = 550e920a : f4b57d6c
Round 14 = f4b57d6c : a04b7445
Round 15 = a04b7445 : 330f217b
Round 16 = 4e930825 : 330f217b
after combination : 4e9382533f217b
* 암호문 : bbf26166928b4210
DES Decryptioning...
Round 16 = 330f217b : a04b7445
Round 15 = a04b7445 : f4b57d6c
Round 14 = f4b57d6c : 550e920a
Round 13 = 550e920a : c243c737
Round 12 = c243c737 : cf1388ad
Round 11 = cf1388ad : 88bcd59b
Round 10 = 88bcd59b : cd0a39f3
Round 9 = cd0a39f3 : 9d1adbf3
Round 8 = 9d1adbf3 : 2ca846dd
Round 7 = 2ca846dd : 6b8df762
Round 6 = 6b8df762 : 6755e21f
Round 5 = 6755e21f : f7acf476
Round 4 = f7acf476 : 8c646118
Round 3 = 8c646118 : d827dbc3
Round 2 = d827dbc3 : 0
Round 1 = ff007f : 0

* 복호문 : 1111111111111110
```

```
*평문 입력 : 1111111111111111
*비밀키 입력 : 22234512987ABB23
DES Encryptioning...
Round 1 = 0 : d827db43
Round 2 = d827db43 : 9c744111
Round 3 = 9c744111 : fcbaa181
Round 4 = fcbaa181 : 24d7fcbe
Round 5 = 24d7fcbe : 9c03697a
Round 6 = 9c03697a : be8f1332
Round 7 = be8f1332 : 56d95b85
Round 8 = 56d95b85 : 33a9bc8f
Round 9 = 33a9bc8f : f72a06a8
Round 10 = f72a06a8 : 2db85f6d
Round 11 = 2db85f6d : 53c65265
Round 12 = 53c65265 : b09bb584
Round 13 = b09bb584 : fb08b73c
Round 14 = fb08b73c : d53d8d15
Round 15 = d53d8d15 : 73fe85a4
Round 16 = c4eec83d : 73fe85a4
after combination : c4eec83d73fe85a4
* 암호문 : 89b07b35a1b3f47e
DES Decryptioning...
Round 16 = 73fe85a4 : d53d8d15
Round 15 = d53d8d15 : fb08b73c
Round 14 = fb08b73c : b09bb584
Round 13 = b09bb584 : 53c65265
Round 12 = 53c65265 : 2db85f6d
Round 11 = 2db85f6d : f72a06a8
Round 10 = f72a06a8 : 33a9bc8f
Round 9 = 33a9bc8f : 56d95b85
Round 8 = 56d95b85 : be8f1332
Round 7 = be8f1332 : 9c03697a
Round 6 = 9c03697a : 24d7fcbe
Round 5 = 24d7fcbe : fcbaa181
Round 4 = fcbaa181 : 9c744111
Round 3 = 9c744111 : d827db43
Round 2 = d827db43 : 0
Round 1 = ff00ff : 0

* 복호문 : 1111111111111111
```

DES는 avalanche effect와 completeness 효과를 가진다. Avalanche effect는 평문(또는 키)에서 작은 변화가 완전히 다른 암호문이 생성되는 것을 말한다. 위에 실행 결과 화면을 보면 평문의 마지막 bit를 0에서 1로 변환한 후 2개의 평문에 대해서 DES 함수를 실행한 결과이다. 2개의 실행 결과 암호문을 보면 단지 bit가 1개 변경되었지만 암호문이 완전히 달라진 것을 확인할 수 있다. 이러한 변화를 avalanche effect라고 한다. Avalanche effect를 통해 악의적인 해커가 선택 암호문 공격을 어렵게 만들 수 있다. 또한 DES의 각 라운드 함수는 P-박스와 S-박스를 통해 diffusion과 confusion을 수행하고, 이러한 결과가 강력한 completeness 효과를 가져다 주는 것을 위에 실행 결과로 확인할 수 있다.

5. 직접 구현한 DES 암호를 사용하여 교재 Chapter 6의 [예제6.8]를 그대로 재현해 보고 그 결과를 상세히 분석하시오.



DES는 2^56개의 키 집합 중에서 대표적인 4개의 취약키를 가진다. 취약키는 패리티 제거 연산 이후에, 모두 0이거나 모두 1이거나 절반은 0이고 절반은 1를 구성한다. 위에 실행결과로 사용된 비밀키는 절반은 1이고 절반은 0으로 구성된 취약키이다. 위에 실행결과로 사용된 취약키를 round key generator에 넣으면 parity drop을 통해서 0000000인 56비트의 actual key가 생성된다. 위에서 생성한 actual key을 가지고 DES의 라운드 함수에서 사용하는 round key를 만들어서 암호화를 진행하면 Ek(Ek(P))=P 과 같은 결과를 야기한다. 위에 실행결과를 보면 평문과 취약키를 넣어서 암호화를 진행하고 생성된 암호문을 다시 DES에 취약키와 넣으면 초기에 입력한 평문을 얻을 수 있다. 즉 해커가 암호문을 가로챈 후 두 번 복호화해서 결과가 입력된 암호문과 같으면 해커는 암호화에 사용된 키를 알아낼 수 있다. 키를 알아내면 암호문에서 평문을 얻을 수 있으므로

DES가 깨지게 되는 것을 위에 실행 결과로 확인할 수 있다.

6. 위의 과제를 수행하면서 개인적으로 느꼈던 점을 자유롭게 서술하시오.

이번 과제를 수행하면서 블록 암호에 대해서 확실하게 이해할 수 있었다. 본 강의를 수강하기 전에는 네트워크 보안이라는 강의명처럼 네트워크 상에서 정보를 보안하는 해킹 방법에 대해서 강의를 듣는 줄 알았다. 하지만 강의를 들으면서 네트워크 상에 전달되는 데이터를 해커로부터 보호하기 위해 데이터를 암호화하는 과정과 복호화 과정에 대해서 이해할 수 있었고, 첫번째로 배우는 블록 cipher인 DES를 통해서 평문으로부터 암호문을 생성하는 방법에 대해 이해할 수 있었다. DES를 이해하고, 구현하면서 DES가 가지는 confusion과 diffusion 성질을 단순한 구조로 풀어내는 방법을 보고 많은 것을 생각할 수 있었다. DES의 상세한 구현을 하면서 bit 단위의 연산들을 처음 접할 때는 bit의 이동을 이해하기 어려웠지만 반복해서 분석하고, 테스트해본 결과 어떻게 bit들이 합쳐지고 쪼개지고 치환되고 대치되는지 확인할 수 있었다. DES 과제를 통해서 블록 암호에 대해서 명확하게 이해하는 계기가 되었다.