DES 개념 및 절차 설명

DES는 Feistel Network 구조를 이용하고, block 단위로 암호화를 실행하는 Block Cipher 알고리즘 이다.

DES는 AES의 개발 전 까지 널리 사용하던 암호화 알고리즘이다. Single DES는 1990년대 컴퓨팅 능력의 발전으로 공격이 성공한 적이 있었고, single Des는 현재, 복호화 알고리즘에만 사용되기를 권장하며, single DES알고리즘을 여러번 사용한 Double Des , Triple Des 가 존재 한다.

## 단어 설명

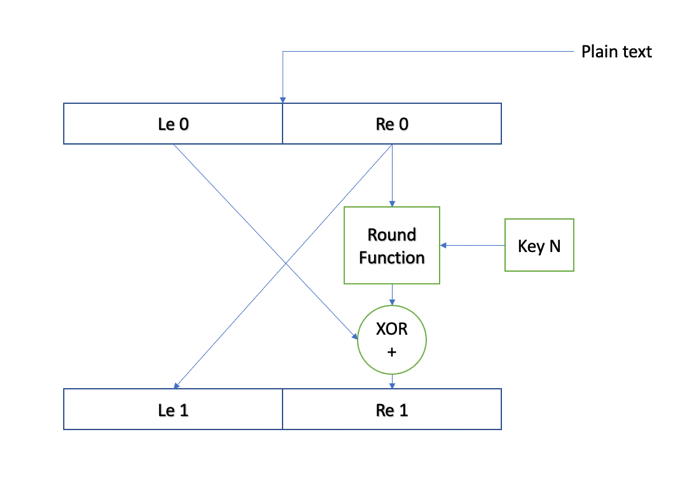
**Block Cipher**

Block cipher는 stream Cipher와 달리 bit, byte, word단위로 순차적으로 암호화 하는 방식이 아니라. 특정 비트의 수의 집합인 Block을 한번에 처리한다. 예를 들어, 11101000 이와 같은 8비트의 byte를 strem Cipher를 통해 bit단위로 key인 00110000과 xor 한다면, 11011000으로 변환 될 것이다. 이는 각 bit별로 연산이 이루어졌는데, block Cipher의 경우 11101000, 11101000, 11101000, 11101000, 11101000, 11101000, 11101000, 11101000,11101000, 11101000, 11101000, 11101000, 이 와 같은 평문이 존재하면 일정한 크기로 잘라낸 후 암호화 알고리즘을 적용한다. 특히 한번만 암호화 과정을 거치는 것이 아니라, 반복적으로 암호화 과정을 거쳐 암호문이 return된다.

**Feistel Network(Cipher)**

Feistel 구조는 substitution(치환), permutation(순열)을 반복하여 암호화를 실행한다. 페이스텔 구조는 반복이라는 특징으로 인해 Diffusion(분산, 확산), Confusion (혼란)이라는 특성을 가지게 된다.

* Diffusion
  + 통계학적인 평문 구조를 광범위한 암호문의 통계로 소멸시킨다. 원문 숫자가 많은 암호문 값에 영향을 미치도록 한다.
* Confusion
  + 암호문의 통계와 암호화 키 값 사이의 관계를 가능한 한 복잡하게 만든다. 비록 공격자가 암호문을 탈취하여도, 암호문을 만드는데 키가 사용된 방법은 너무 복잡해서 키를 추론하는 것을 어렵게 만든다.



(Feistel Cipher Structure)

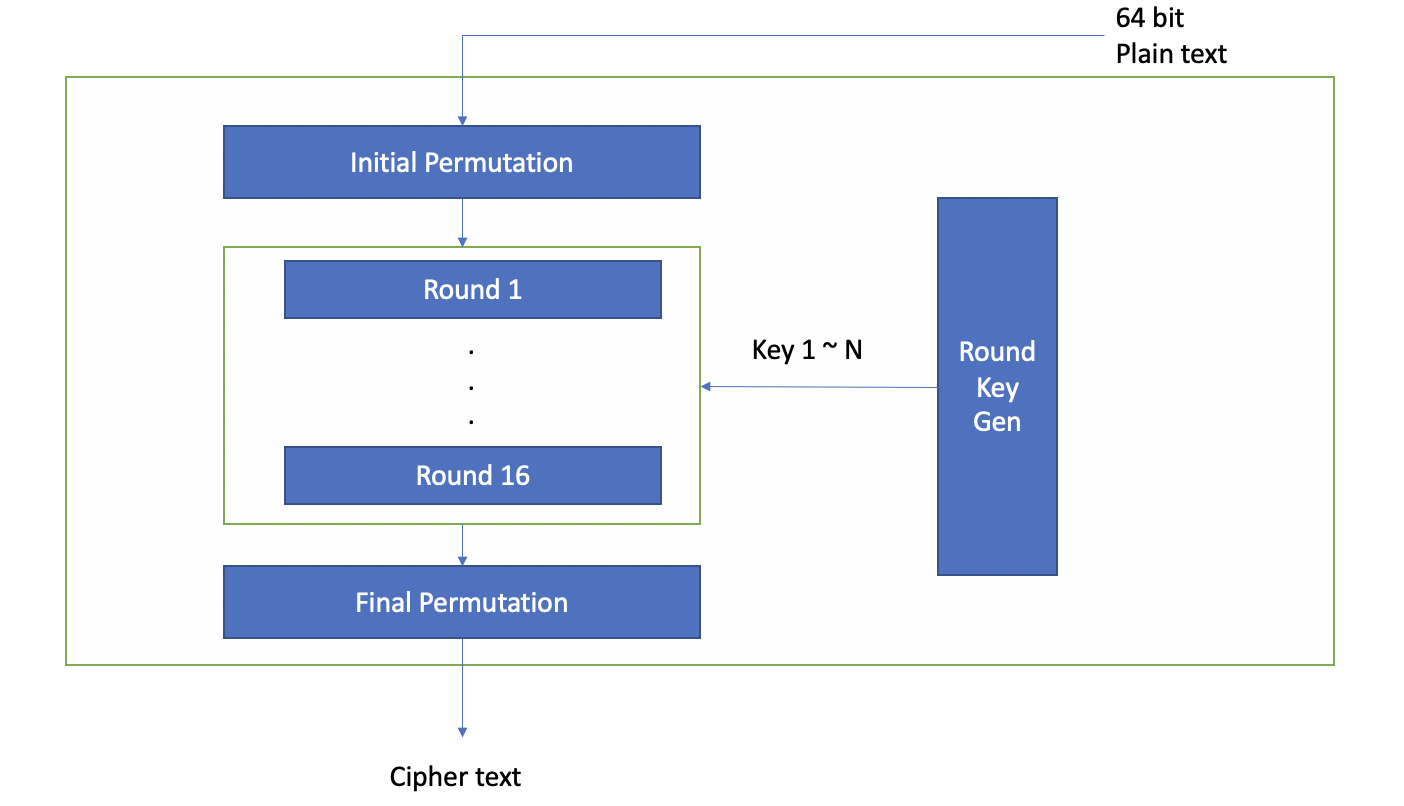
**How to Encrypt plain text using Feistel Network Structure**

1. 암호화 대상을 절반으로 나누어 L과 R비트로 나눈다.
2. R비트를 새로 만드는 암호문의 L bit로 지정한다.
3. R비트와 키와 함께 라운드 함수를 실행한다.
4. 라운드 함수의 결과를 암호화 전 L 비트와 XOR한다.
5. 4의 결과를 암호문의 R비트로 지정한다.
6. 5의 결과로 나온 암호문을 위의 1 ~ 5의 과정을 지정된 라운드 만큼 반복한다.

**How to Decrypt cipher text using Feistel Network Structure**

1. 복호화 대상의 비트를 절반으로 나누어 L과 R비트로 나눈다.
2. L비트를 복호화한 평문의 R비트로 지정한다.
3. L비트를 라운드 함수에 넣고 Key와 함께 실행한다.
4. 라운드 함수 실행 결과를 암호문의 R비트와 XOR한다.
5. 4의 결과를 평문의 L비트로 지정한다.
6. 5의 결과로 나온 평문을 위의 1 ~ 5 과정을 지정된 라운드 만큼 반복한다.

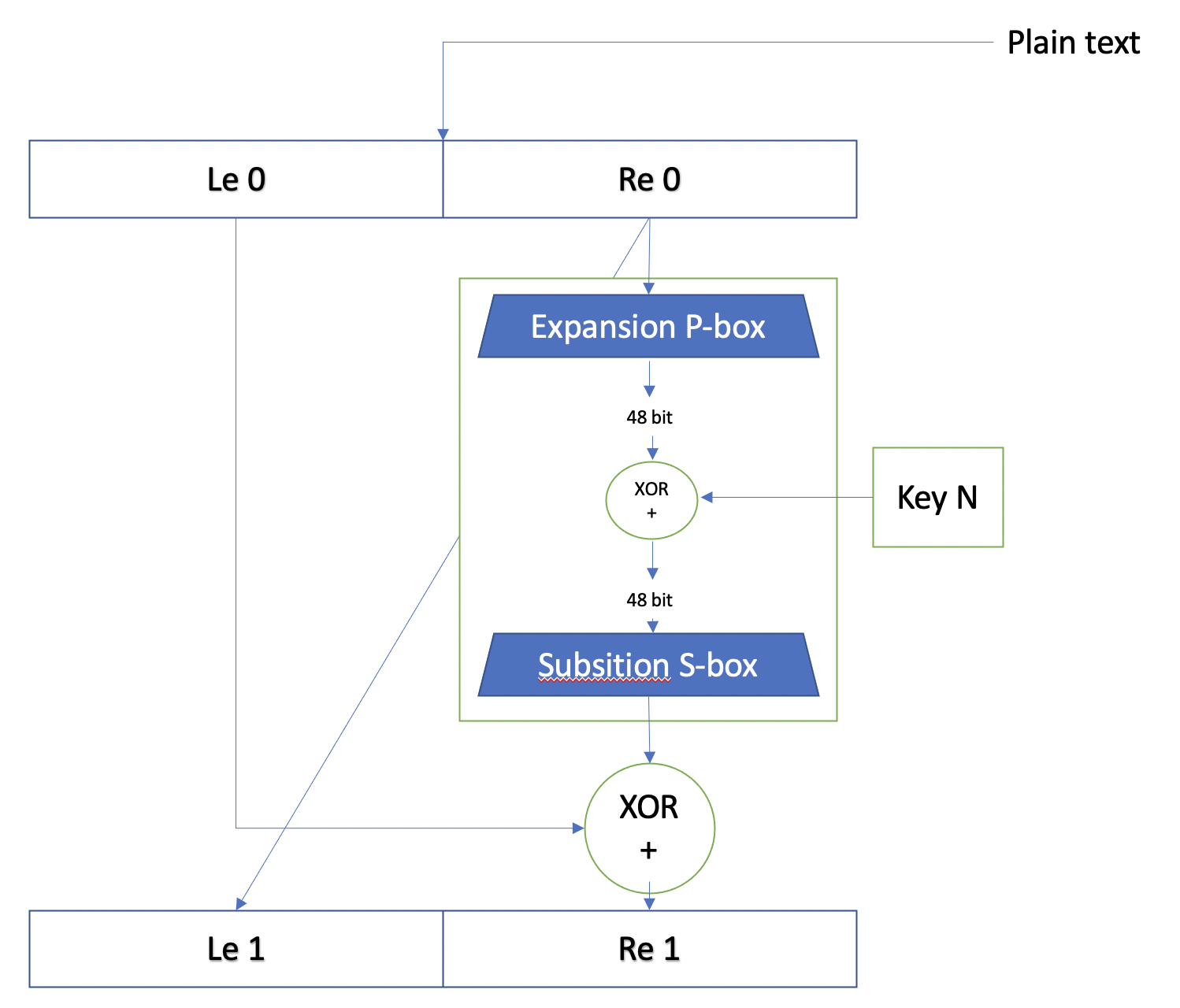
**Data Encryption Standard**

1977년에 개발된 DES는 2001년 AES의 개발 전까지 널리 사용되었다. DES는 64bit의 key중 56bit는 round function에 사용되는 key이며, 나머지 8bit는 키의 오류를 검증하는 Parity bit이다. 

(페이스텔 구조를 사용하는 DES 암호화 방법)

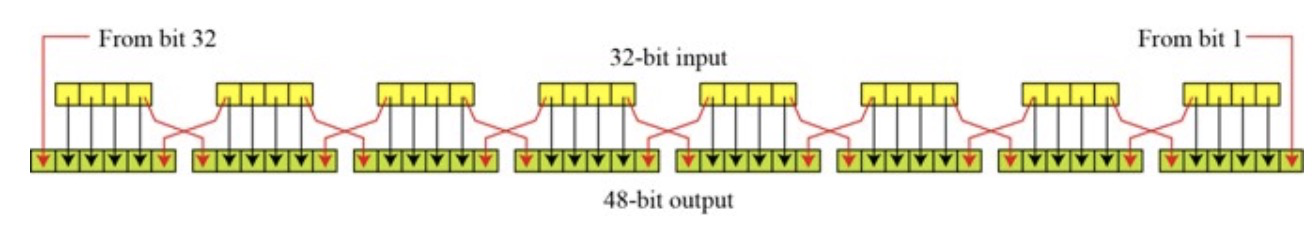
위의 페이스텔 구조에서 추가된 것은 Round 함수를 실행하기전에, text의 bit를 정해진 bit table에 맞추어 자리이동 하는 것과 라운드 함수가 지정된 라운드를 모두 끝낸 후 다시 한번 자리이동 하는 것이다. Permutation이 진행 되는 동안에는 치환이 이루어지지 않는다.

**Round function in DES**



Round Function매개변수로 암호화 대상인 R bit와 N번째 Round Key가 넣어지게 된다. R 비트는 P box를 통해 32bit에서 48비트로 확장이 이루어지게 되며, 이 48비트를 이용하여 Key의 48bit와 XOR연산을 하게 된다. 이후 S-box 내부에서 확장된 암호문 48bit가 32bit로 압축된다. 이후 32bit의 좌우 를 스왑하는 것으로 Round Function이 종료되고 평문의 L비트와 XOR하여 암호문의 R비트로 지정되게 된다 .

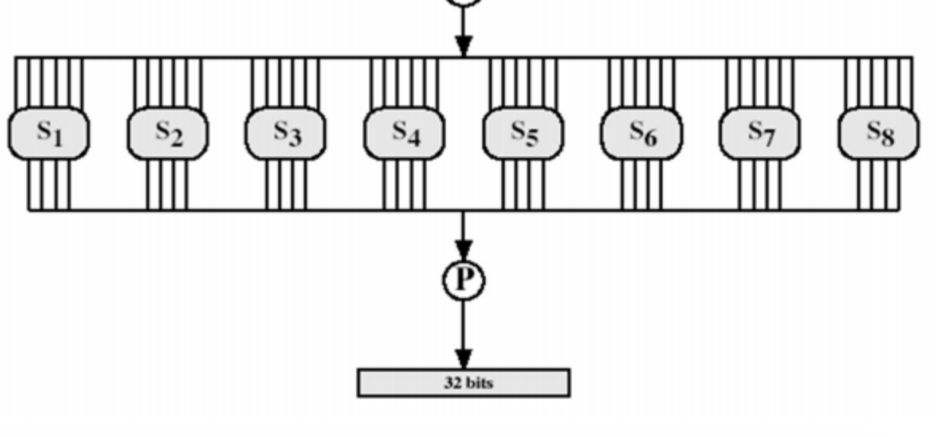
**Expansion P-box**



(출처 : <https://developer-mac.tistory.com/52> )

P-box 에서는 평문 32비트를 8구간을 4비트로 나누어 8구간으로 6비트의 2^1~4 자리에 삽입한다. 6비트 중 1,5비트는 해당 6비트의 1번과 4번 비트의 원문의 왼쪽, 오른쪽 비트를 삽입한다. 만들어진 48비트를 이용하여, 48비트의 key와 XOR 연산을 실행한다.

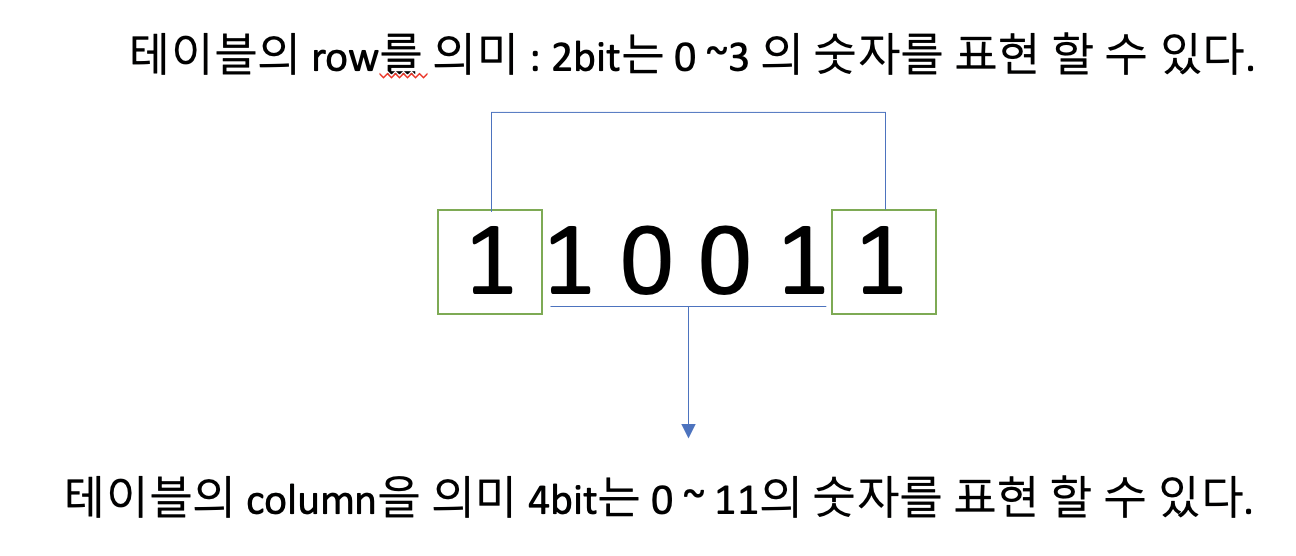
**Subsition S-box**

~~~~

S-box내부에서는 48bit로 확장된 암호를 32bit로 줄이는 압축을 담당한다. ~~48bit의 2^2~4 비트는 기존 원문에 key와 함께 xor연산되어 있기 때문에 0,5번째 비트를 제외한 비트를 추출하여, 32bit의 암호문을 만들어 낸다. 이후 32bit을 좌우로 나누어 좌, 우로 swap 하는 과정을 거친다. 이후 마지막으로 원문의 L비트와 round function의 결과물을 xor 연산하여 암호문의 R비트로 지정한다.~~

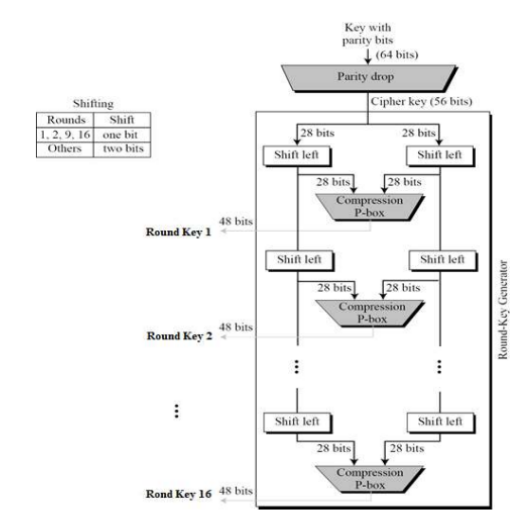
(수정)

각 S-box에는 치환을 위한 Table이 존재한다. S-box의 매개변수로 전달된 6개의 비트는 다음과 같이 테이블에서 colum과 row를 의미하게 된다.



6bit에서 4bit로 압축이 일어나기 때문에 테이블에서는 0 ~ 11까지의 숫자로 랜덤하게 배치되어 있고, 위의 6bit가 의미하는 row와 column을 찾아서 해당 숫자로 bit를 치환한다.

**Key Generator**

****

(출처: Cryptography Just for Beginners])

Key Genrerator는 64bit의 키를 매개변수로 입력 받는 것으로부터 시작 된다. PC-1이라는 Parity drop box에 64비트를 넣게 되면, 8의 배수 (8,16,24,32,40,48,56)의 bit는 제외되어 Cipher Key가 된다.

이후 56bit의 cipher key는 절반으로 split 되어 28bit씩 가지게 된다. 각 28bit는 비트를 왼쪽으로 이동 시키는 left shift 연산을 실행하게 되는데, 그림 왼쪽 위의 설명에 나와 있듯이 1,2,9,16 round는 1bit 씩 shift하며 3,4,5,6,7,8,9,10,11,12,13,14,15라운드는 2 bit씩 shift 하게 된다. 이후 이 56bit의 key는 pc-2 box의 매개 변수로 사용된다.

Pc-2 box에서도 shift된 bit중 8의 배수를 제거하여 56bit에서 48비트로 압축된다. (이 과정에서 혹자는 9의 배수라고 하는 사람도 있다.)