정보보호론 #3

대칭키 암호시스템의 이해

Prof. Byung Il Kwak

Review

- Hash function
- Message Authentication

Contents

- 대칭키 암호화 방식
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES) 소개

CONTENTS

- 대칭키 암호화 방식

대칭키 암호화 방식

- 암호학 소개
  - 1980년 이전 암호학은 주로 군이나 정보기관에서 일반 문서의 비밀(기밀성)을 제공하기 위하여 사용됨
    - 컴퓨터와 인터넷의 급속한 발전으로 현재 암호학은 컴퓨터 시스템과 관련된 거의 모든 분야의 보안 메커니즘을 구현하기 위한 필수적인 부분이 되고 있음
      - 예) 웹에서 서비스를 제공받기 위한 인증 및 접근 제어, 기업 등의 기밀 유출을 방지하기 위한 데이터 유출방지(Data Loss Prevention, DLP) 시스템, 소프트웨어와 컨텐츠를 보호하기 위한 저작권 관리(Digital Right Management, DRM) 기술 등에 널리 사용
      - 또한, 모바일 장치에서의 개인정보보호를 위한 목적에도 암호가 사용되고 있음

대칭키 암호화 방식

- 암호학 소개
  - 암호학(Cryptology)은 암호기법(Cryptography)과 암호분석(Cryptanalysis) 분야로 나뉘어짐
    - 암호기법을 뜻하는 영문표기인 Cryptography는 그리스어로 “비밀(secret)”을 의미하는 Kryptos와 “쓰다(Write)”를 의미하는 gráphō의 합성어로 어원적 의미는 “Secret writing”
  - 현대 이전에 암호기법은 메시지(Message)를 감추어 읽지 못하도록 하기 위한 목적, 즉 메시지의 기밀성(Confidentiality)을 제공하기 위하여 사용됨

대칭키 암호화 방식

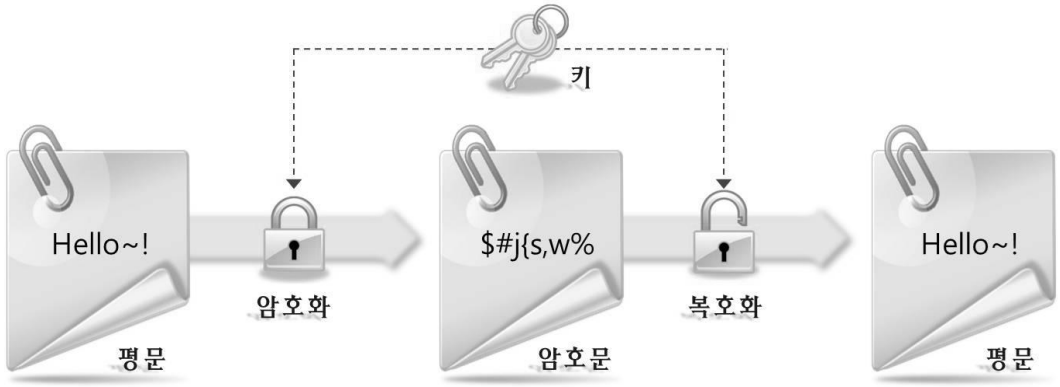
- 암호학 소개
  - 암호기법에서 '**암호화(Encryption 혹은 Encipherment)**'는 알고리즘(Algorithm)을 사용하여 메시지(정보) 의미를 알 수 없는 형태로 변형시키는 과정
    - 암호화를 위하여 사용되는 알고리즘을 특별히 **암호화 알고리즘(Encryption Algorithm)**, 혹은 **암호(Cipher)**라고 함
    - 여기서 암호화 이전의 메시지는 **평문(Plaintext)**, 암호화된 후의 메시지는 **암호문(Ciphertext)라고 함**
    - 즉, 암호 알고리즘은 평문을 입력 받아 암호문을 출력하며, 평문은 항상 의미 있는 문장일 필요 없음
    - 암호화를 두 번 수행하는 경우, 첫 번째 생성된 암호문이 두 번째 암호화의 입력으로 사용되기도 하기 때문

대칭키 암호화 방식

- 암호학 소개
  - 암호화의 반대 과정, 즉 복호화(Decryption 혹은 Decipherment)는 알고리즘을 사용하여 암호문을 평문으로 바꾸는 과정 (암호문 -> 평문)
  - 암호화/복호화 알고리즘을 합쳐서 암호 알고리즘(**Cryptographic Algorithm**), 혹은 암호시스템(**Cryptographic Systems / Cryptosystems**)라고 함
    - 평문을 암호문으로 바꾸거나 암호문을 평문으로 바꾸어서 본래의 메시지를 읽기 위해서는 **특별한 비밀이 필요 (키 - Key)**
    - 평문을 암호문으로 바꿀 때 사용하는 키를 **암호화 키(Encryption Key)**, 암호문을 평문으로 바꿀 때 사용하는 키를 **복호화 키(Decryption Key)**라고 함

대칭키 암호화 방식

- 암호학 소개
  - 알고리즘은 어떠한 문제를 해결하기 위한 유한(finite)한 단계적인 절차나 방법을 말함
  - 키 등의 사전정보 없이 암호문에서 평문을 복호화하는 기법을 **암호분석(Cryptanalysis)** 이라 하며, 과거 암호분석은 정보기관이나 범죄집단에 의해서 수행되는 것으로 생각되었지만, 현재는 대부분 연구기관에서 연구되고 있으며 하나의 학문분야로 자리잡음



대칭키 암호화 방식

- 암호 알고리즘
  - 암호 알고리즘은 다음과 같이 세 개의 알고리즘 $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{D})$로 구분됨
    - 키 생성 (*Key Generation*) 알고리즘 $\boldsymbol{G}$
      - 키 생성 알고리즘 $\boldsymbol{G}$ 는 가능한 키들의 **집합 $\boldsymbol{K}$**에서 **암호화 키 $\boldsymbol{k_1}$**와 **복호화 키 $\boldsymbol{k_2}$**를 선택함
      - 이 때, **집합 $\boldsymbol{K}$**를 **키 공간(Key Space)**이라 함
      - 한 개의 키가 **집합 $\boldsymbol{K}$**에서 선택될 확률은 키 공간의 분포에 따라 상이하지만, 일반적으로 균일분포를 따르는 확률 $\frac{\mathbf{1}}{|\boldsymbol{K}|}$을 가지는 난수 (Uniformly Distributed Random Number)를 선택함
      - ex) 4 bit 길이의 키를 사용하는 경우 키 공간은 집합 {0, 1, . . . , 15}이며, 균일분포를 따라 이 중 하나의 값이 키로 선택될 확률은 $\frac{1}{16}$으로 볼 수 있음

대칭키 암호화 방식

- 암호 알고리즘
  - 암호 알고리즘은 다음과 같이 세 개의 알고리즘 $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{D})$로 구분됨
    - 암호화 (Encryption) 알고리즘 $\boldsymbol{E}$
      - **암호화 알고리즘** $\boldsymbol{E}$는 **암호화 키** $\boldsymbol{k_1} \in \boldsymbol{K}$를 사용하여 **평문** $\boldsymbol{m}$을 입력으로 받아 암호문 $\boldsymbol{c}$를 출력함

$$\boldsymbol{E_{k_1}(m) = c}$$

    - 복호화 (Decryption) 알고리즘 $\boldsymbol{D}$
      - 복호화 키 $k_2 \in K$를 사용하여 암호문 $c \in C$를 복호화하면 메시지 $m \in M$을 얻어낼 수 있음

$$\boldsymbol{D_{k_2}(c) = m}$$

대칭키 암호화 방식

- 암호 알고리즘
  - 암호 알고리즘은 다음과 같이 세 개의 알고리즘 $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{D})$로 구분됨
    - 암호 알고리즘이 올바르게 설계되었다면 **평문** $\boldsymbol{m}$의 암호문을 복호화 했을 때 **평문** $\boldsymbol{m}$을 얻을 수 있어야 함.
      - 즉, $\boldsymbol{D_{k_2}(E_{k_1}(m)) = m}$ 수식을 만족해야 함.

대칭키 암호화 방식

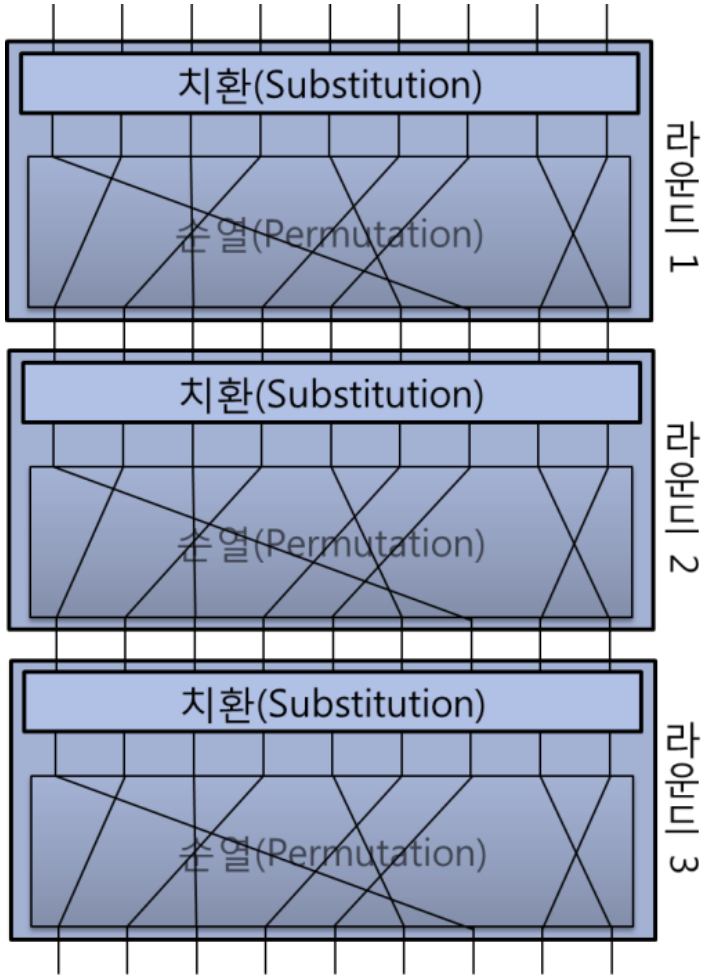
- 암호 알고리즘
  - 암호 알고리즘은 다음과 같이 세 개의 알고리즘 $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{D})$로 구분됨
    - 이러한 상황에서, 암호 알고리즘은 **암호화 키**와 **복화화 키**가 1. **같은 경우**와 2. **다른 경우**로 구분됨
      - 암호 알고리즘이 **동일한 암호화 키와 복호화 키를 사용하는 경우**, 즉 $\boldsymbol{k_1 = k_2}$인 경우, **대칭키 알고리즘(Symmetric Key Algorithm)**이라 함
      - 만약 암호 알고리즘이 **다른 키$(\boldsymbol{k_1 \neq k_2})$를 사용하는 경우**, **비대칭키 알고리즘(Asymmetric Algorithm)** 또는 **공개키 알고리즘(Public Key Algorithm)**이라 함
    - 대칭키 알고리즘은 암호화되는 평문의 크기에 따라 **블록 암호(Block Cipher)**와 **스트림 암호(Stream Cipher)**로 분류됨

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 암호문을 생성(암호화)할 때, 사용하는 키와 암호문으로부터 평문을 복원할 때 사용하는 키가 동일한 암호 시스템
    - 대칭키 암호는 혼돈(confusion)과 확산(Diffusion) 성질을 이용하여 평문을 암호화함
      - **Confusion: 키와 암호문과의 관계를 감추는 성질**
        - 현대 블록암호는 혼돈을 위해 치환 (Substitution)을 사용
      - **Diffusion: 평문과 암호문과의 관계를 감추는 성질**
        - 평문 한 비트의 변화가 암호문의 모든 비트에 확산됨
        - 주로 평문과 암호문의 통계적 성질을 감추기 위해 사용함
        - DES에서는 여러 번의 순열(Permutation)을 사용하여 확산 성질을 만족하고, AES에서는 좀 더 발전된 형태인 MixColumn을 사용함

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 곱 암호(Product Cipher)
    - 샤논은 혼돈과 확산을 함께 사용하면 안전한 암호를 설계할 수 있다고 제안
    - 라운드(Round)라 불리는 부분이 여러 번 반복되는 구조
    - 한 라운드에는 치환과 순열이 사용됨



대칭키 암호화 방식

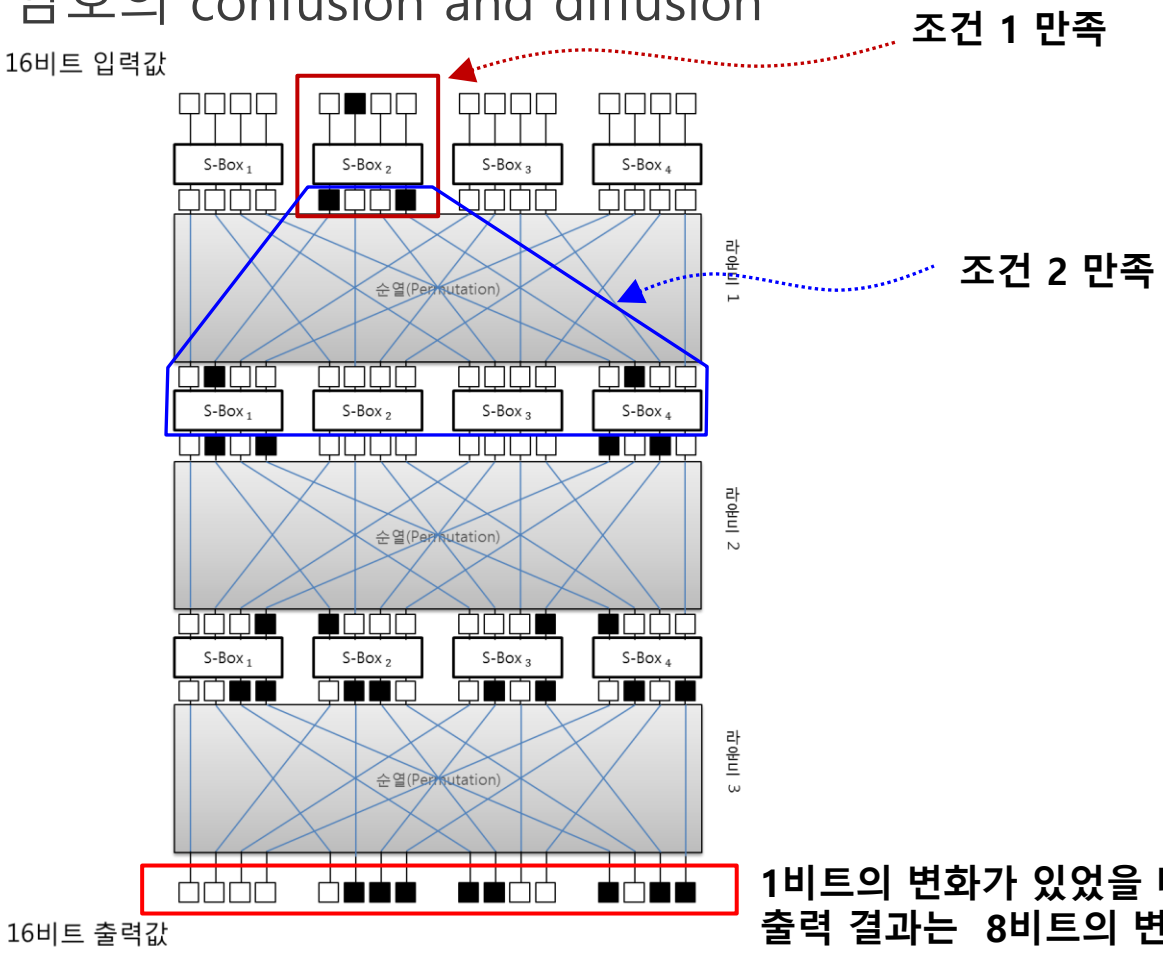
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 1973년 미국의 연방 표준국(National Bureau of Standards, NBS) DES 공모
    - IBM은 자사의 루시퍼(Lucifer)를 제출
    - 미 연방 표준국은 1977년 루시퍼를 수정하여 DES로 선정
      - FIPS PUB 46
      - 국가안보국(National Security Agency, NSA)는 루시퍼에서 사용된 64비트 키를 56비트로, S-Box의 형태도 변형
      - considerable controversy over its security
    - most widely used block cipher in world
    - 64-bit 데이터 블록, 56-bit 키
      - 최대 2^{56}번의 계산 필요 -> 현재 2^{80} 정도의 계산이 안전
      - 현재 3-DES(triple DES)와 AES 사용

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 곱 암호의 형태
      - Design principle : **S-Box** (Substitution Box)
      - **조건 1**. 1 bit가 다른 두 입력 값이 S-Box에 입력되었을 때 각 입력 값에 대응되는 두 출력 값은 2 bit 이상이 달라야 함
      - **조건 2**. 하나의 S-Box에서 출력된 비트들은 다음 라운드에서 모든 S-Box들의 입력 값으로 확산되어 들어 갈 수 있도록 순열을 설계해야 함
      - Note: 블록 사이즈가 2^n인 곱 암호가 충분한 혼돈과 확산 성질을 만족하기 위해서는 $(n-1)$라운드 이상으로 구성

대칭키 암호화 방식

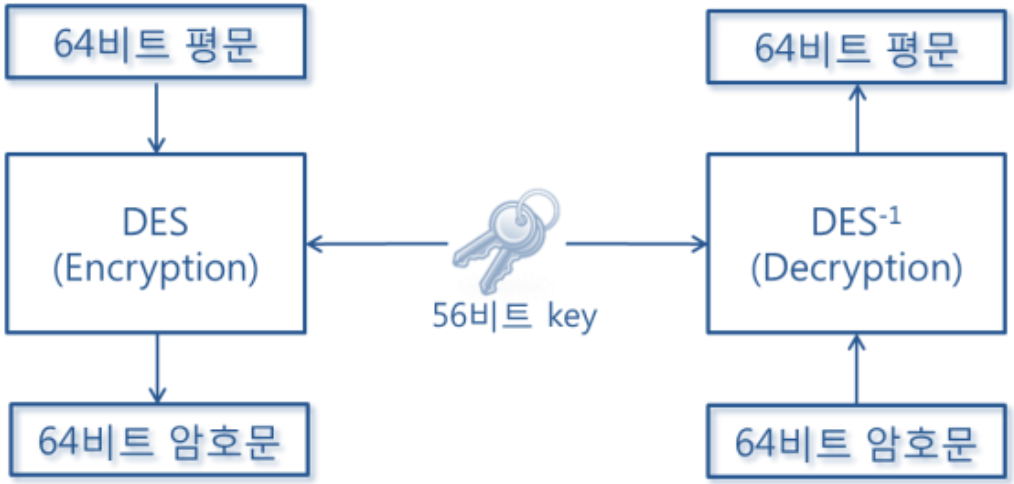
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 곱 암호의 confusion and diffusion



1비트의 변화가 있었을 때,
출력 결과는 8비트의 변화로 이어짐

대칭키 암호화 방식

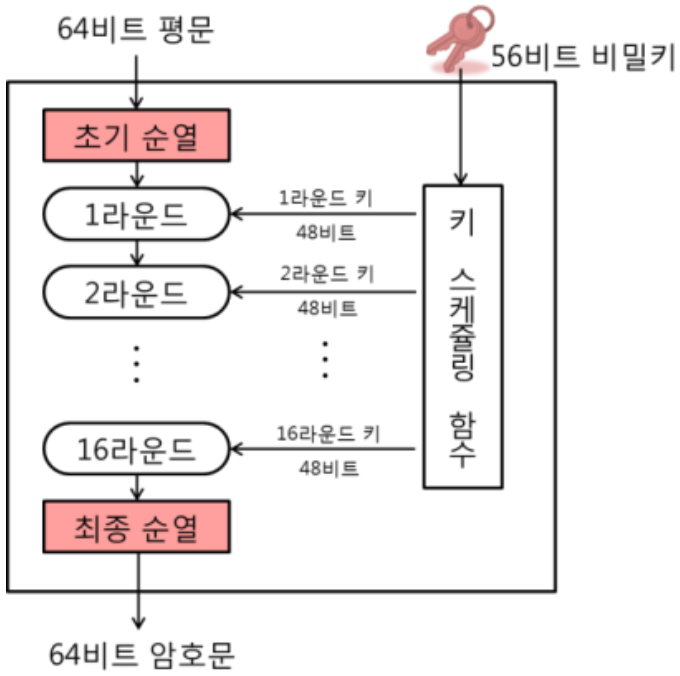
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)



- DES는 대칭키 블록 암호이며, 암호화와 복호화 과정에서 서로 같은 키를 사용하며 블록 단위로 암호화와 복호화를 수행
  - 한 블록의 크기는 64 비트이며 키 길이는 56비트
  - 즉, 64비트의 평문을 56비트의 키로 암호화하여 64비트의 암호문을 생성함. 또한, 동일한 키를 사용하여 64비트 암호문을 64비트의 평문으로 복호화함.

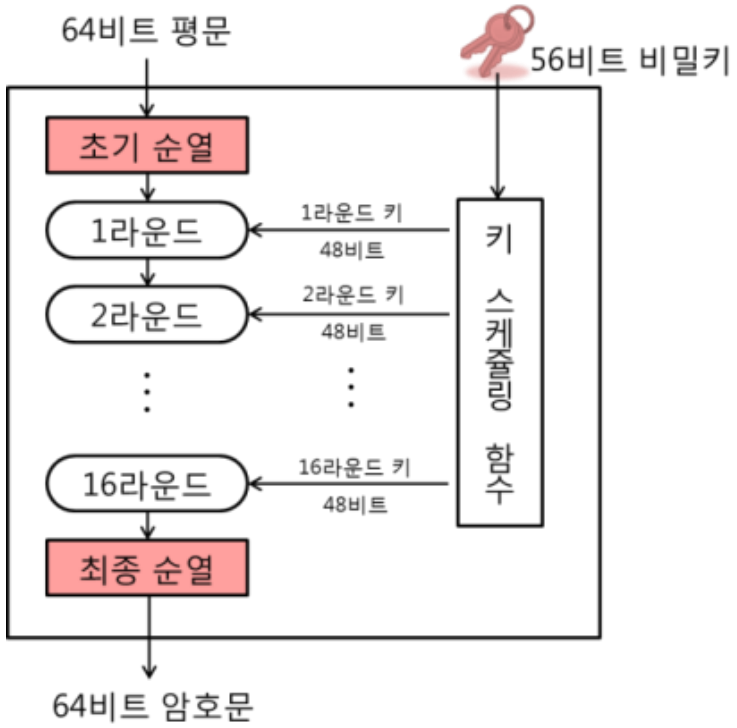
대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - DES는 많은 현대 암호들과 같이 라운드가 여러 번 반복되는 곱 암호 구조를 갖고 있음.



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 초 기 순 열 IP (Initial Permutation), 최종 순열 FP (Final Permutation), 16개의 라운드 그리고 키 생성함수로 구성됨
    - 최종 순열은 초기 순열과 역관계이며 IP^{-1}로 나타냄
    - 키 생성 함수는 56비트의 대칭키를 사용하여 각 라운드별로 48비트의 상이한 라운드 키를 생성함



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 곱 암호의 두 가지 형태
      - 페이스텔(Feistel) 암호 : *가역(Invertible)*요소와 *비가역(Non-Invertible)* 요소 모두를 사용
        - 암호화와 복호화 과정이 동일
      - 비페이스텔(Non-Feistel) 암호 : 가역 요소만 사용

이전의 상태로 돌아 갈 수 있음

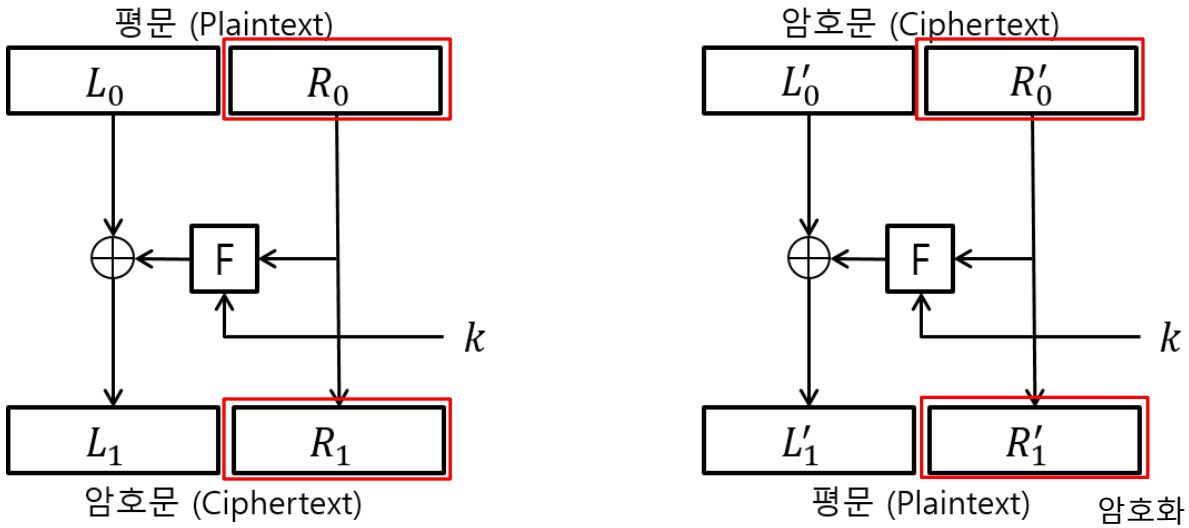
이전의 상태로 돌아 갈 수 없음

    - Note : 전단사함수 $f: X \rightarrow Y$에 대하여 Y에서 X로의 역관계가 존재하면 이를 역함수 (Inverse Function)라고 하며 $f^{-1}: Y \rightarrow X$로 나타냄.
      - 가역함수는 바로 역함수가 존재하는 전단사함수(1:1대응함수)를 의미

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 페이스텔(Feistel) 암호

1라운드 페이스텔 구조



암호화 및 복호화가 되지 않아, 안전하지 않음

암호화 : $c = p \oplus (F(k, R_0) \,||\, 0)$

복호화 : $c \oplus (F(k, R'_0) \,||\, 0) = p$

$= p \oplus (F(k, R_0) \,||\, 0) \oplus (F(k, R'_0) \,||\, 0)$

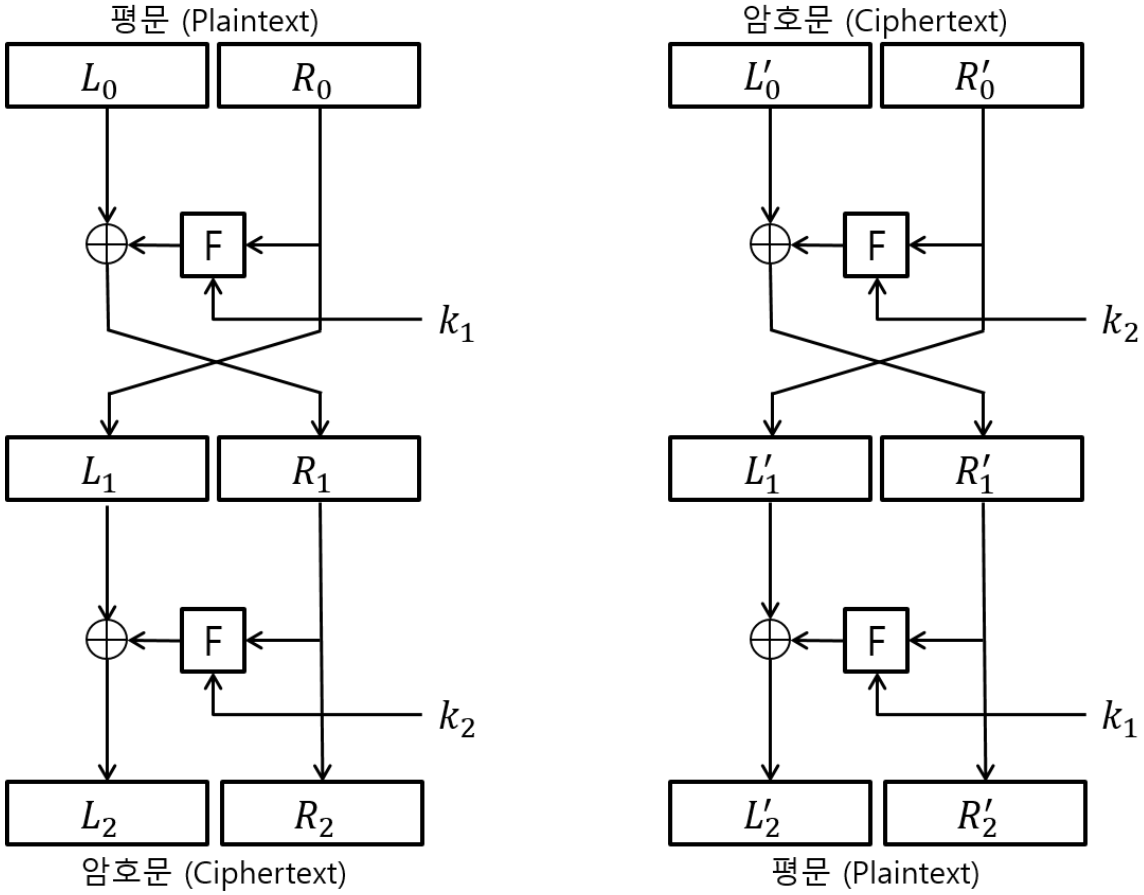
$= p \oplus (F(k, R_0) \,||\, 0) \oplus (F(k, R_0) \,||\, 0)$

$= p \oplus 0 = p$

대칭키 암호화 방식

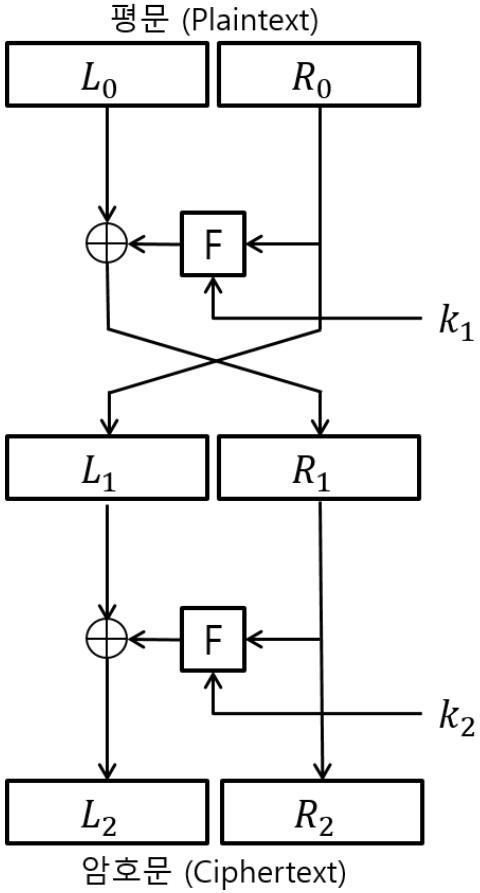
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 페이스텔(Feistel) 암호

2라운드 페이스텔 구조



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 페이스텔(Feistel) 암호



2라운드 페이스텔 구조

암호화

$$L_1 = R_0$$

$$R_1 = L_0 \oplus F(k_1, R_0)$$

$$L_2 = L_1 \oplus F(k_2, R_1)$$

$$R_2 = R_1$$

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 페이스텔(Feistel) 암호

복호화

$$L_1' = R_0' = R_2 = R_1$$

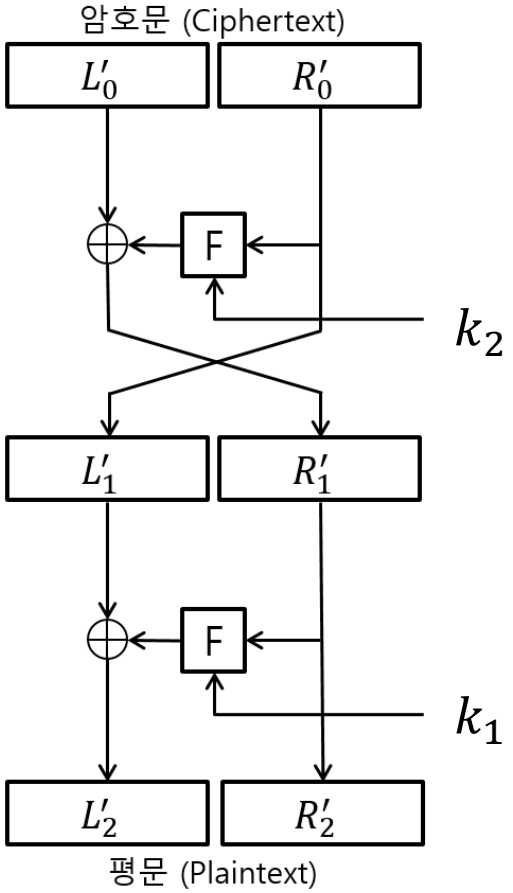
$$R_1' = L_0' \oplus F(k_2, R_0') = L_2 \oplus F(k_2, R_1)$$

$$= L_1 \oplus F(k_2, R_1) \oplus F(k_2, R_1) = L_1 = R_0$$

$$\boldsymbol{L_2'} = L_1' \oplus F(k_1, R_1') = R_1 \oplus F(k_1, R_0)$$

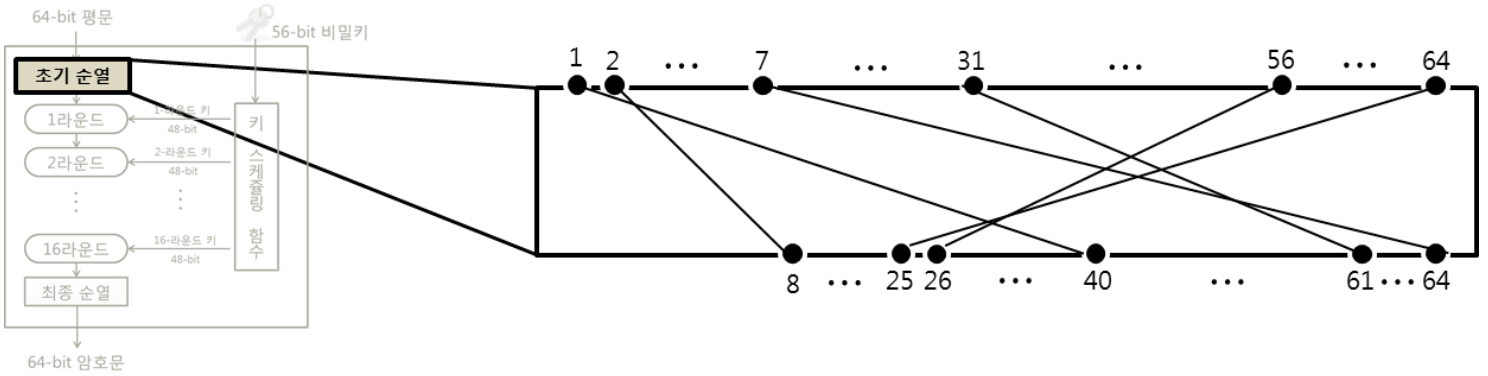
$$= L_0 \oplus F(k_1, R_0) \oplus F(k_1, R_0) = \boldsymbol{L_0}$$

$$R_2' = R_1' = \boldsymbol{R_0}$$



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 초기 순열 IP / 최종 순열 FP

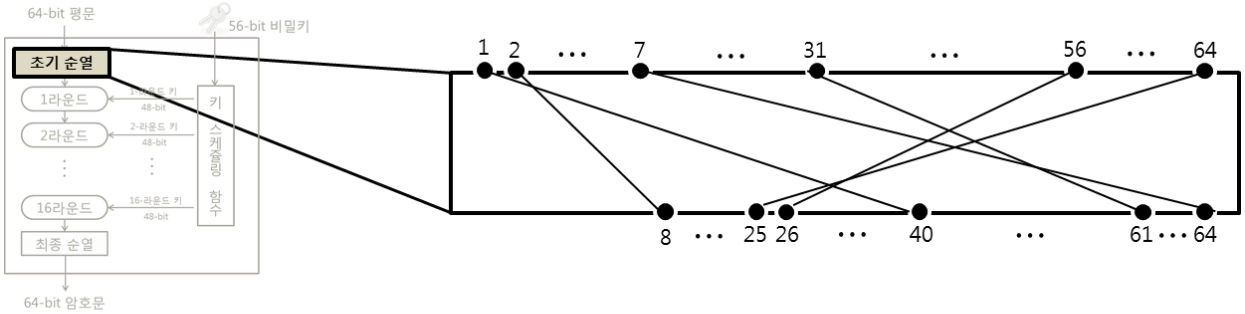


초기 순열 (Initial Permutation)

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - **초기 순열 IP** / 최종 순열 FP
      - 64비트의 평문이 입력되면 가장 먼저 실행되는 과정이 초기 순열 IP(Initial Permutation) 과정이며, 단순하게 비트 단위의 자리바꿈이 일어남



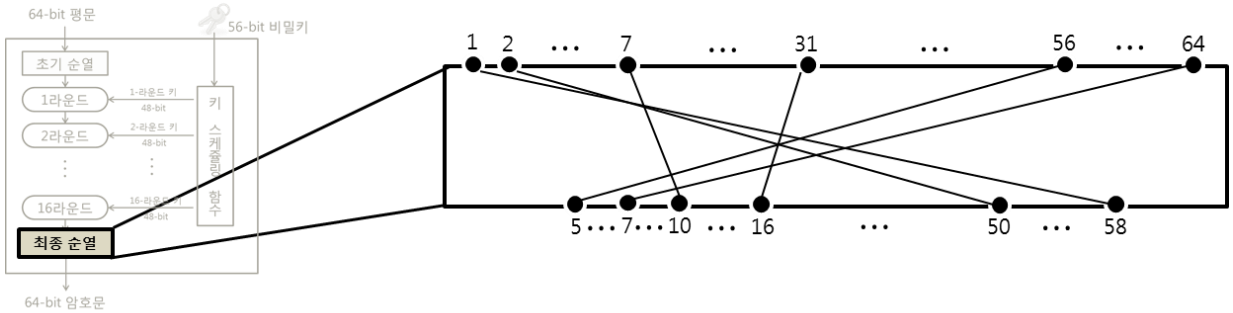
** 초기 순열
- 왼쪽에서 오른쪽으로, 위에서 아래로 읽음
- 첫 번째 행의 왼쪽에서 여덟 번째 값인 02_2는 입력 값의 2_2번째 비트를 출력 값의 8_2번째 비트로 자리바꿈 하라는 의미

초기 순열 (Initial Permutation)

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 초기 순열 IP / **최종 순열 FP**
      - DES의 암호화 과정 중 가장 마지막에 실행되는 과정으로, 초기 순열과 역관계이며, 초기 순열에서 바뀌었던 자리를 원래 위치로 되돌려 놓는 순열 (IP^{-1}로 표기)



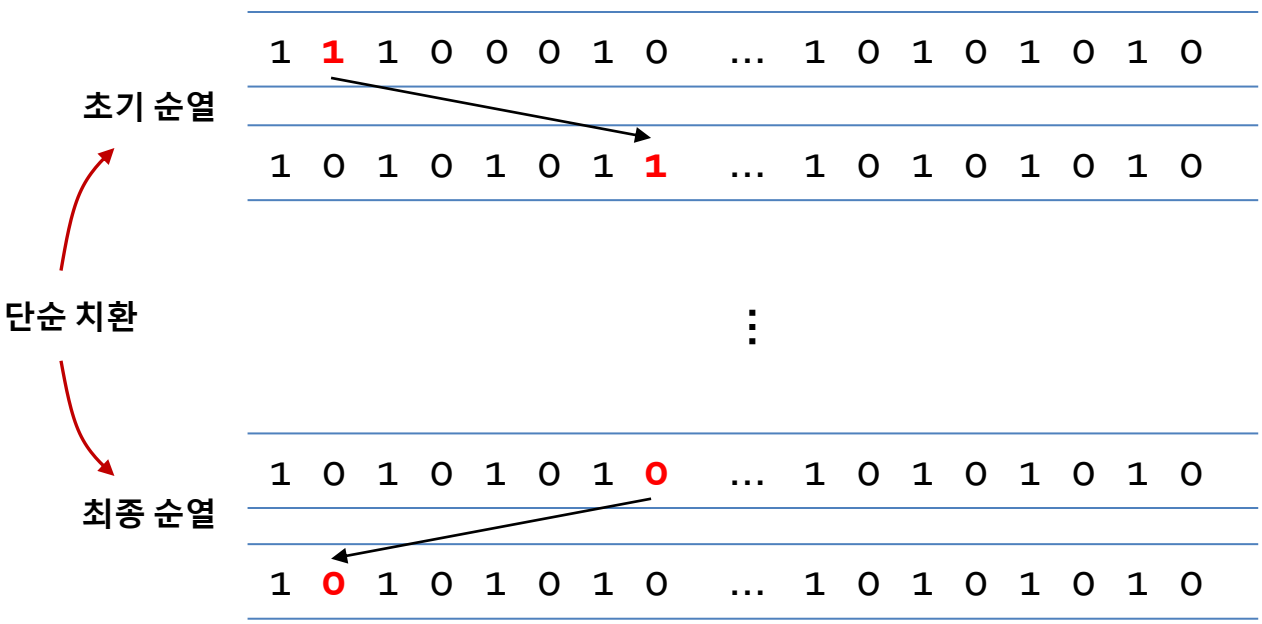
** 최종 순열
- 왼쪽에서 오른쪽으로, 위에서 아래로 읽음
- 첫 번째 행의 왼쪽에서 두 번째 값인 08은 입력 값의 8번째 비트를 출력 값의 2번째 비트로 자리바꿈 하라는 의미

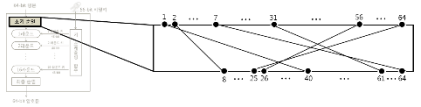
최종 순열 (Final permutation)

40	08	48	16	56	24	64	32	39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30	37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28	35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26	33	01	41	09	49	17	57	25

대칭키 암호화 방식

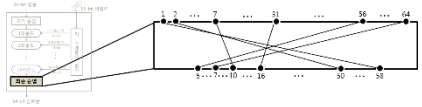
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - **초기 순열 IP / 최종 순열 FP**
      - 초기 순열과 최종 순열은 단순 치환이기 때문에, DES의 안전성을 증가시키지 못함.





초기 순열 (Initial Permutation)

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

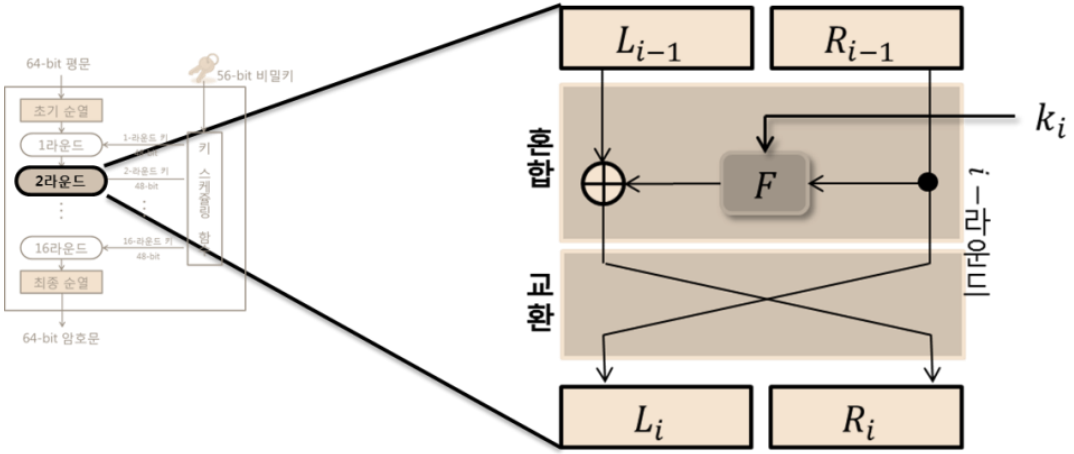


최종 순열 (Final permutation)

40	08	48	16	56	24	64	32	39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30	37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28	35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26	33	01	41	09	49	17	57	25

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수)
      - DES는 총 16라운드로 구성되어 있으며, 각각의 라운드는 혼합(Mixer)과 교환(swapper)과정으로 구성됨
      - 마지막 16라운드에서는 교환 부분이 존재하지 않고, 혼합 과정만 존재함



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수)
      - **혼합(Mixer) 과정**
        - 오른쪽 32비트 부분과 그 라운드에 해당하는 키를 이용해 F함수 값을 계산한 후, 왼쪽 32비트 부분과 그 값을 XOR 연산을 수행

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

      - **교환(Swapper) 과정**
        - XOR 연산을 통해 계산된 값을 오른쪽으로, 처음 입력된 값의 오른쪽 32비트를 왼쪽으로 위치를 바꿈 $(L_i = R_{i-1})$

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

대칭키 암호화 방식

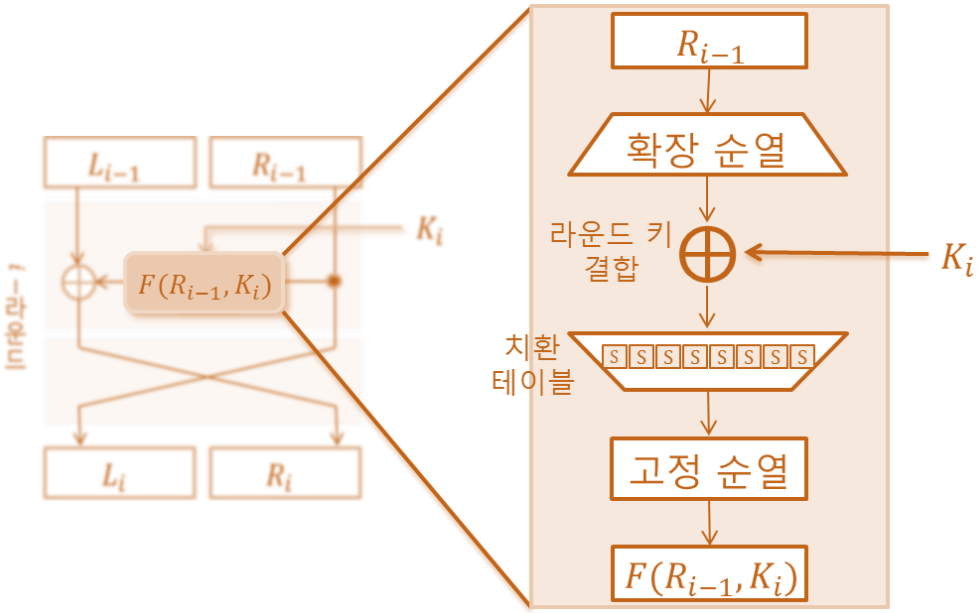
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수)
      - 혼합 과정에서 사용되는 ***F*함수**를 ***DES 함수***라고 하며, 이 함수는 각 라운드 입력 값의 오른쪽 32비트인 $\boldsymbol{R_{i-1}}$와 해당 라운드의 키 $\boldsymbol{k_i}$를 이용해서 32비트의 값을 출력함
      - 라운드의 키 $\boldsymbol{k_i}$는 48비트이며 이 라운드 키를 생성하는 방법은 키 스케쥴링(Key Scheduling) 함수 부분에서 설명

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수)



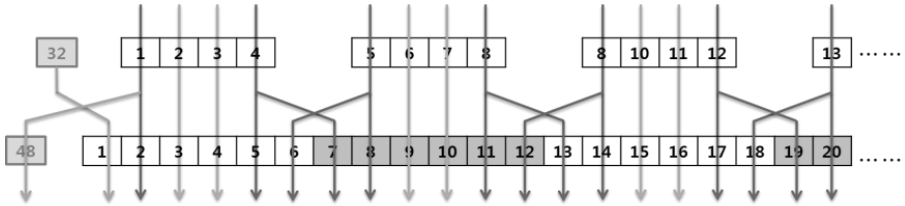
- 페이스텔 구조를 갖고 있는 DES에서 F함수는 앞에서 언급한 혼돈과 확산 성질을 구현한 부분
  - 확장 순열(Expansion P-Box)
  - 라운드 키와 XOR로 결합하는 부분
  - 8개의 치환 테이블(S-Box)
  - 고정 순열(Straight P-Box)

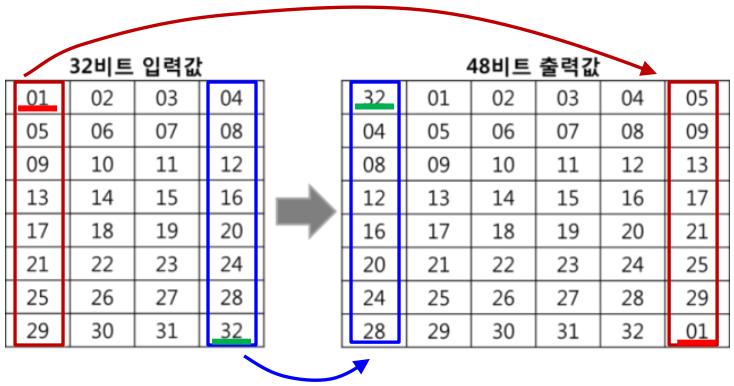
대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 확장순열(Expansion P-Box)
      - 확장 순열 과정에서는 32비트 입력 값을 비트 단위로 순서를 자리바꿈과 함께 출력 값의 크기를 48비트로 확장하는 역할
      - 입력 값을 확장시키는 방법
        - 32비트의 입력 값을 4비트씩 8개의 부분으로 분할
        - 정해진 규칙에 따라 각각의 4비트를 6비트로 확장
          - 왼쪽 하단의 테이블은 32비트 입력 값이 4비트씩 나뉘어져서 위 행부터 채워져 있으며, 오른쪽 하단 테이블은 왼쪽 테이블의 각 행, 즉 4비트가 6비트로 확장된 결과를 나타냄
          - 확장 방식은, i번째 행의 4비트 값 맨 앞과 맨 뒤에 한 비트씩을 앞에 추가되는 비트는 $(i-1)$번째 행의 4비트 값 중, 4번째 비트를 추가하고 뒤에 추가되는 비트는 $(i+1)$번째 행의 4비트 값 중 1번째 비트를 추가
          - 첫 번째 행의 4비트 값의 경우, 맨 앞에 추가되는 비트는 전체 입력 값의 32번째 비트가 되고, 8번째 행의 4비트 값의 경우, 맨 뒤에 추가되는 비트는 전체 입력 값의 1번째 비트가 됨

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 확장순열(Expansion P-Box)





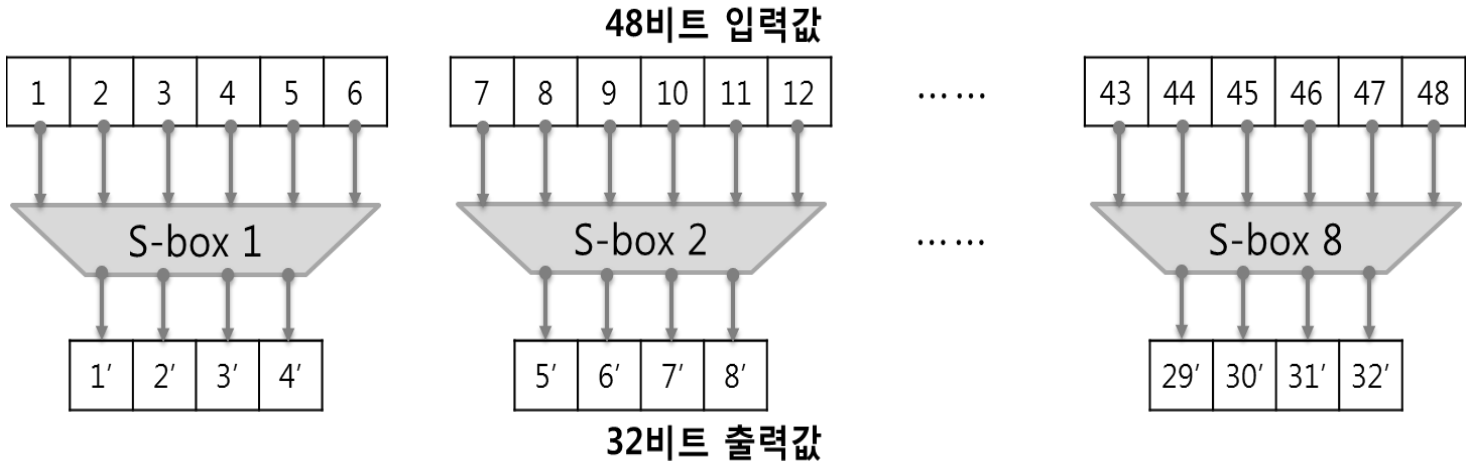
- 왼쪽 하단의 테이블은 32비트 입력 값이 4비트씩 나뉘어져서 위 행부터 채워져 있으며, 오른쪽 하단 테이블은 왼쪽 테이블의 각 행, 즉 4비트가 6비트로 확장된 결과를 나타냄
- 확장 방식
  - **i번째 행의 4비트 값 맨 앞과 맨 뒤에 한 비트씩을 앞에 추가되는 비트는 $(i-1)$번째 행의 4비트 값 중, 4번째 비트를 추가하고 뒤에 추가되는 비트는 $(i+1)$번째 행의 4비트 값 중 1번째 비트를 추가**
  - 첫 번째 행의 4비트 값의 경우, **맨 앞에 추가되는 비트는 전체 입력 값의 32번째 비트가 되고, 8번째 행의 4비트 값의 경우, 맨 뒤에 추가되는 비트는 전체 입력 값의 1번째 비트**가 됨

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 라운드 키 결합(XOR)
      - 확장 순열을 통해 32비트의 입력 값이 48비트로 확장되면 해당 48비트 값은 라운드 키 결합 과정에서 48비트 라운드 키 k_i와 서로 XOR 연산됨
      - 16개의 라운드 키는 DES의 56비트 비밀키에 의해 만들어 지며 이 과정은 키 스케쥴링 함수 부분에서 설명하며, DES 함수에서 라운드 키는 이 과정에서만 사용됨

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 변환 테이블(S-Box)
      - 48비트의 입력 값을 6비트씩 8개의 부분 분할
      - 각 6비트를 순서대로 8개의 S-Box에 입력 수행
      - 각 S-Box는 6비트를 입력 받아 4 비트로 줄여 출력 수행

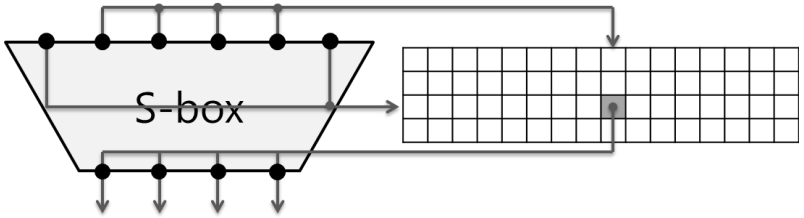


대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 변환 테이블(S-Box)
      - S-Box는 4개의 행과 16개의 열로 이루어진 표이며, 4 비트로 표현될 수 있는 수들로 구성됨 (0~15 사이의 숫자)
      - 입력된 6 비트를 4 비트로 치환하는 규칙
        - **규칙 1.** 입력된 6비트 중 첫 번째와 여섯 번째 비트를 붙여서 이 값으로 행을 결정. 즉, 첫 번째와 여섯 번째 두 비트는 이진수로 00, 01, 10, 11 중의 하나가 되며, 십진수로 0부터 3행 중의 하나를 결정
        - **규칙 2.** 나머지 2번째부터 5번째까지 4 비트는 열을 결정. 즉, 십진수로 0부터 15열 중의 하나를 결정
      - 즉, 6비트가 S-Box에 입력되면 규칙 1과 규칙 2에 의하여 행과 열이 결정되고, 그에 해당하는 표의 숫자가 출력되며, 이 값은 네 비트 값으로 십진수로 0부터 15까지 중 하나의 숫자

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 변환 테이블(S-Box)



100111

Row = 11 = 3

Column = 0011 = 3

➔ 02

S-box 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

S-box 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

. . . .

S-box 8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	10	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	8	13	15	12	09	00	03	05	06	11

대칭키 암호화 방식

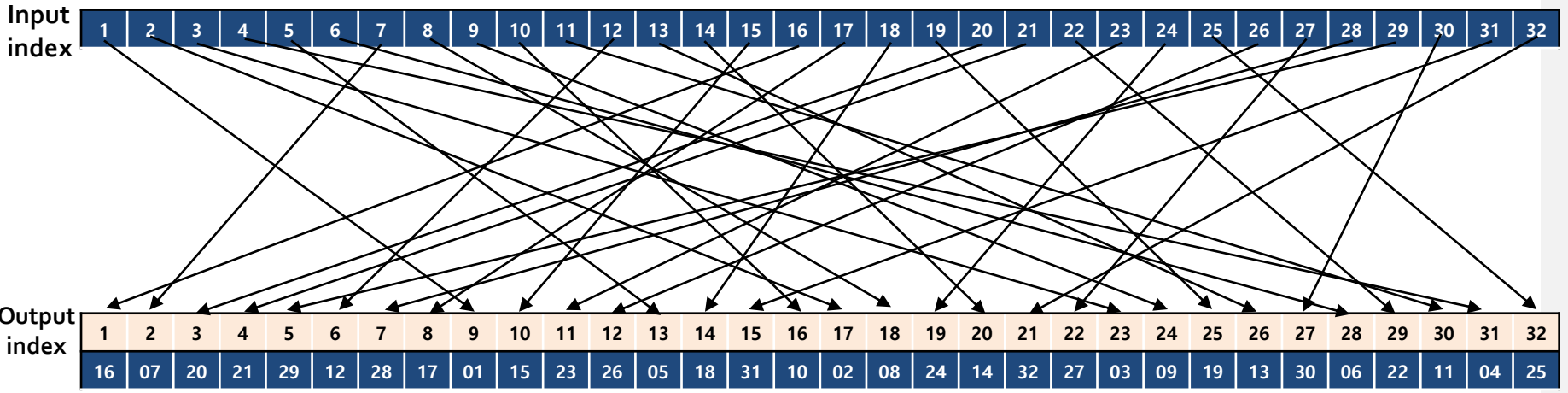
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 고정 순열(Straight Permutation)
      - 고정 순열 과정에서는 치환 테이블 과정에서 출력된 32비트 값을 자리바꿈함
        - 예를 들어, 입력 값의 16번째 비트는 출력 값의 1번째 비트가 됨

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

고정 순열 표(Straight Permutation)

대칭키 암호화 방식

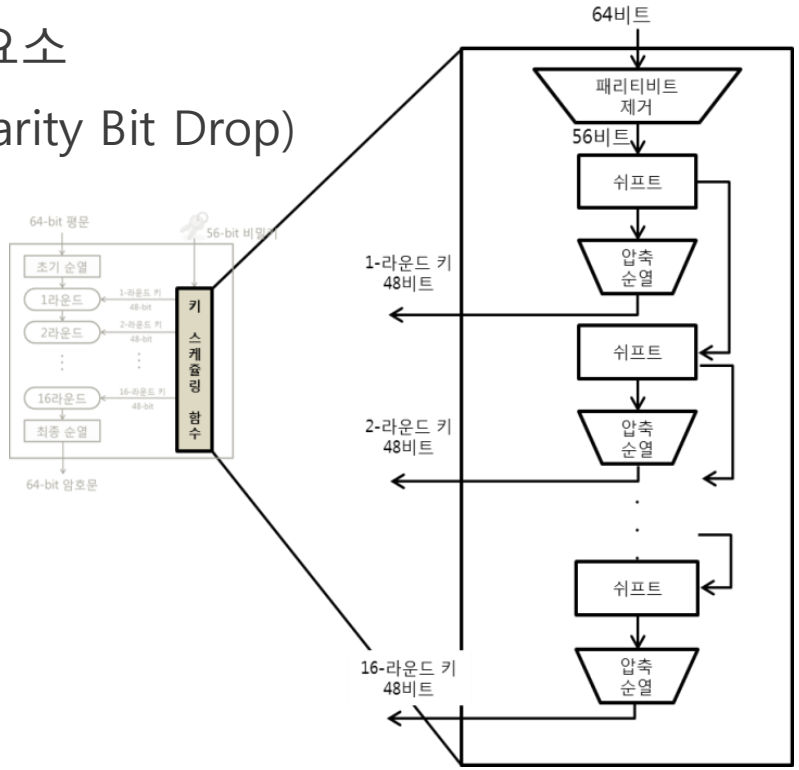
- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 라운드 함수(*F* 함수) - 고정 순열(Straight Permutation)
      - 고정 순열 과정에서는 치환 테이블 과정에서 출력된 32비트 값을 자리바꿈함
        - 예를 들어, 입력 값의 16번째 비트는 출력 값의 1번째 비트가 됨



고정 순열 표(Straight Permutation)

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 키 스케쥴링(Key Scheduling)
      - 함수는 64비트인 비밀키를 입력으로 받아 16개의 48비트 라운드 키를 출력
      - 키 스케쥴링 함수의 구성요소
        - 1) 패리티비트 제거(Parity Bit Drop)
        - 2) 쉬프트(Shift)
        - 3) 압축 순열



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 키 스케쥴링(Key Scheduling)
      - 패리티 비트 제거(Parity Bit Drop)
        - 패리티 비트(Parity Bit)는 데이터의 전달 과정에서 오류 발생 여부를 검사하기 위한 비트를 말하며 한 바이트, 즉 8비트 중 1비트를 말함
        - DES 암호의 64비트 비밀키 중 8개의 패리티 비트(8의 배수 비트-> 8, 16, 24, 32, 40, 48, 56)는 키의 안전성에 기여하지 않기 때문에, 일반적으로 DES 암호는 56비트 비밀키를 사용함(패리티 비트 제거)

패리티 비트 제거 표

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 키 스케쥴링(Key Scheduling)
      - 쉬프트(Shift)
        - 입력으로 들어오는 56비트를 왼쪽 28 비트와 오른쪽 28 비트로 나누어, 각각의 28비트를 1 비트나 2 비트 만큼 왼쪽으로 순환 쉬프트(Cyclic Shift)하는 과정
        - 라운드 1, 2, 9, 16의 경우는 1 비트, 나머지 라운드에서는 2 비트 순환 쉬프트 수행

라운드	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
횟수	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

라운드 별 쉬프트 비트 표

** 처음 64비트에서 패리티 비트를 제거한 56비트와 16번째 라운드 키가 동일하게 됨
이러한 성질은 암호화 과정의 라운드 키를 역순으로 사용하는 DES의 복호화 과정에서 유용

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - 키 스케쥴링(Compression P-Box)
      - 압축 순열(Shift)
        - 56비트의 입력 값을 48비트로 출력하는 과정
        - 예) 56비트의 입력 값 중 14번째 비트를 1번째 비트로 이동하고 32번째 비트를 48번째 비트로 이동을 의미
        - 이렇게 나오는 출력 값은 해당 라운드의 48비트 라운드 키로 사용
          - 8개 비트를 사용하지 않음 (09, 18, 22, 25, 35, 38, 43, 54)

압축 순열 표

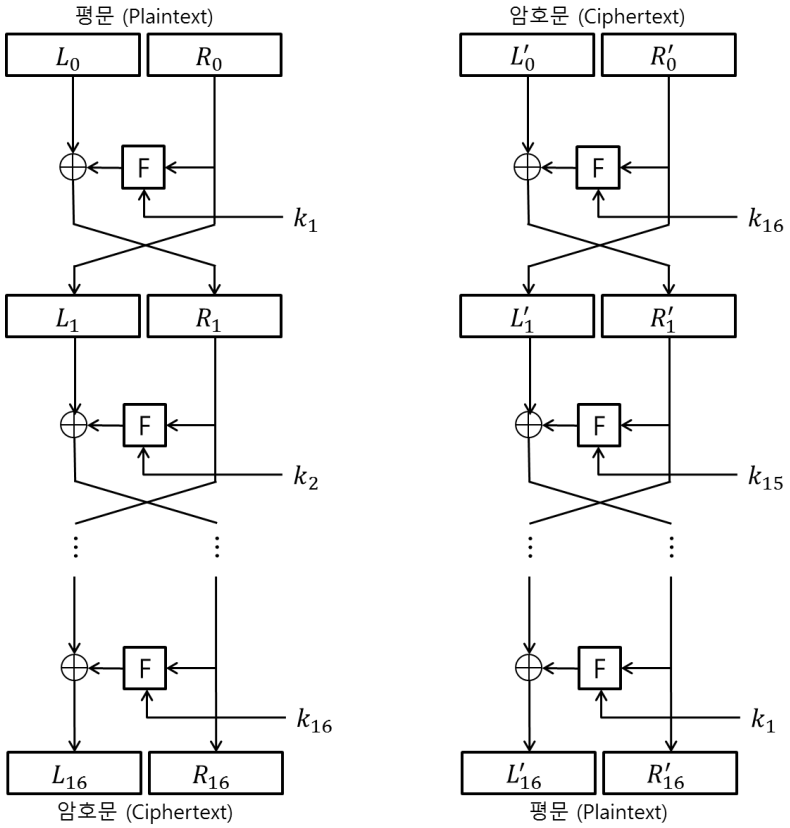
14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - DES 복호화
      - DES는 페이스텔 구조를 지니며 비가역 구성요소를 사용함
      - 비가역 구성요소로는 각 라운드 함수(F함수)에 포함되어 있는 확장 순열 (Expansion P-Box)과 S-Box가 있음
      - 라운드 함수의 라운드 키 결합 과정과 교환(Swapper) 과정은 가역이기 때문에 복호화 시 문제가 없지만, 비가역 구성요소들은 그 자체로 복호화가 되지 않음
      - 앞에서 기술한 것처럼 배타적 논리합(XOR) 연산을 사용해서 이런 문제를 해결
      - 비가역 구성요소들이 들어있는 라운드 함수의 출력 값은 암호화 시 각 라운드 입력 값의 왼쪽 반과 XOR되며, 복호화를 위해서는 라운드 함수의 출력 값을 다시 한번 XOR 연산이 필요
        - 즉, 라운드 함수와 관계 없이 XOR 만으로 복호화 가능

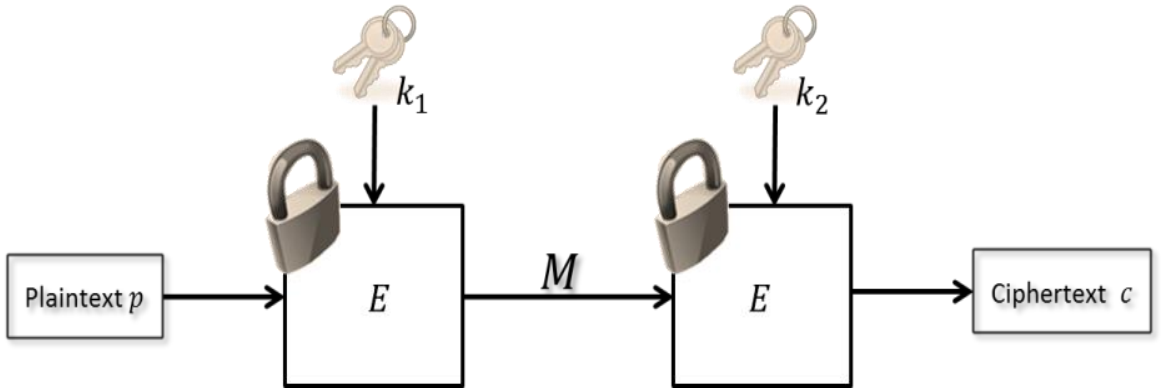
대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - Data Encryption Standard (DES)
    - DES 복호화



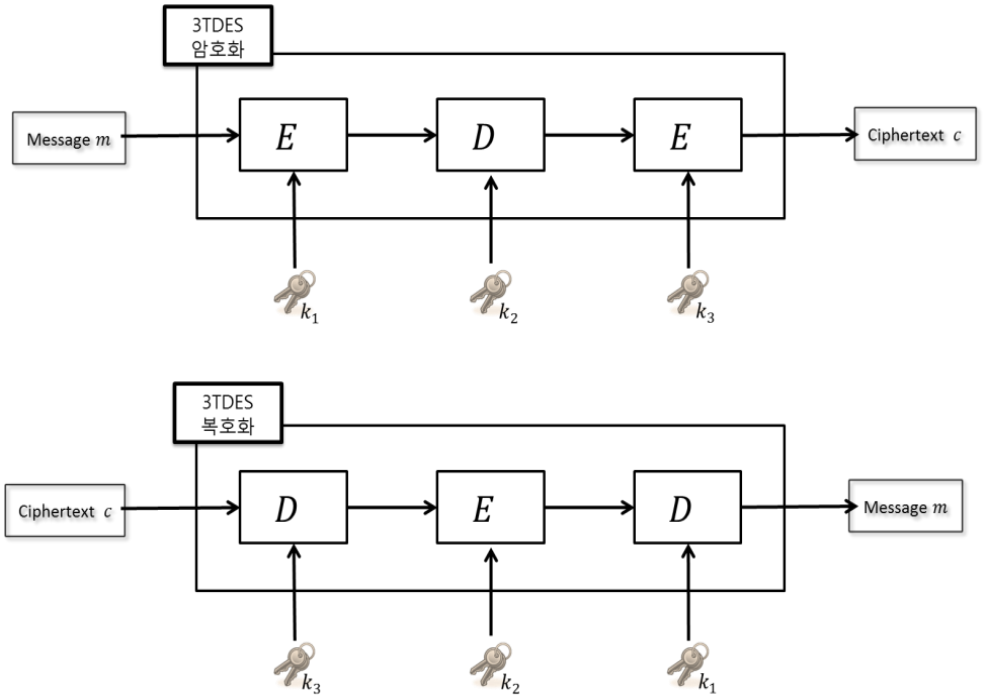
대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 다중 DES
    - 56비트인 DES암호의 짧은 키 길이를 보완하기 위해 DES를 여러 번 사용하는 다중 DES가 제안
    - 2중 DES (Double DES)



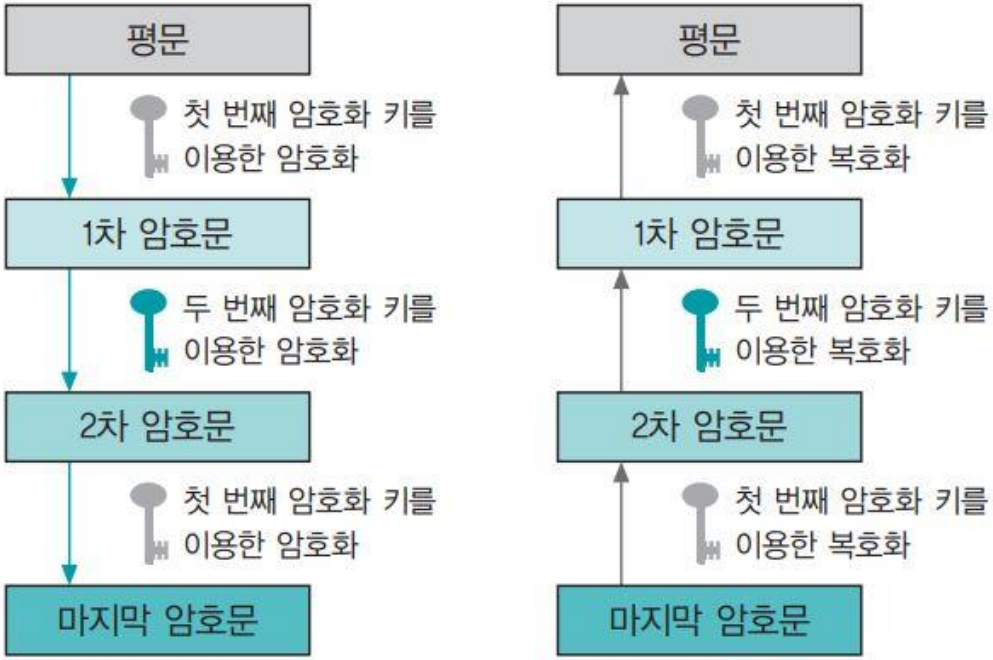
대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 다중 DES
    - 2중 DES (Double DES)
      - 2중 DES의 경우 중간 일치 공격(Meet-in-the-Middle Attack)에 취약
      - 그러한 이유로, 3중 DES (Triple DES)가 제안됨



대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 다중 DES
    - 3중 DES (Triple DES)



CONTENTS

- Advanced Encryption Standard (AES) 소개

대칭키 암호화 방식

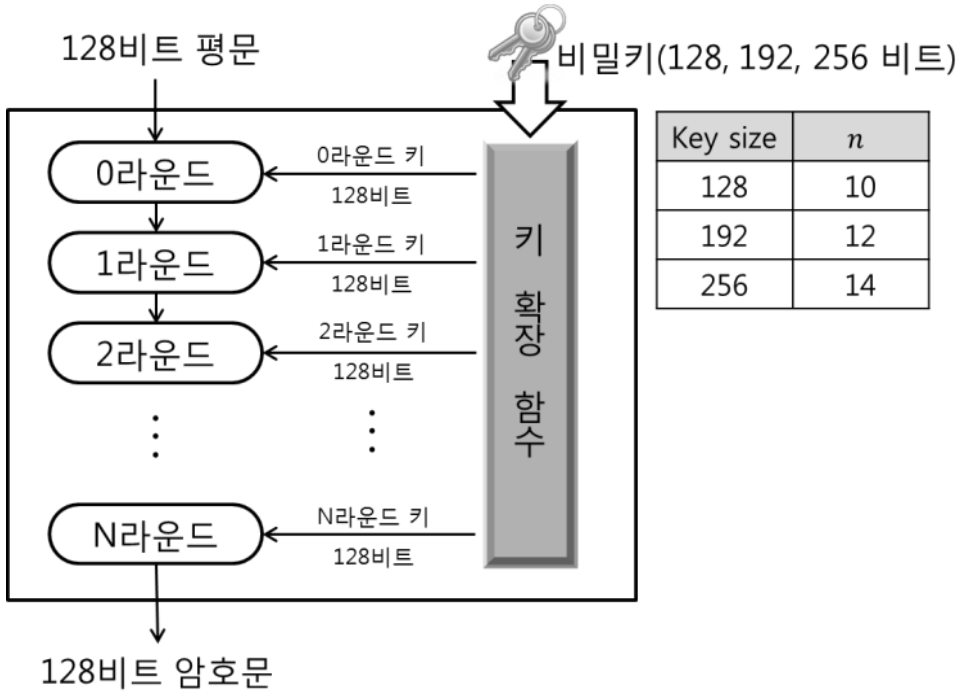
- 대칭키 암호화(Symmetric key)
  - AES (Advanced Encryption Standard)
    - DES의 2^{56}개의 키에 대한 전사적 공격이 가능
      - 1999년 distributed.net 과 Electronic Frontier Foundation이 협력한 공격에서 DES의 비밀키를 22시간 15분만에 찾아냄
    - TDES가 있지만 다음 이유로 NIST에서는 AES 공모
      1. TDES는 DES를 세 번 사용하기 때문에 속도가 느림
      2. DES의 블록 크기인 64 비트는 여러 가지 응용분야에 적합하지 않다. 예로 블록 암호를 이용하여 설계한 해쉬 함수(8장에서 소개)의 경우 64비트의 블록 크기는 해쉬 함수의 안전성에 문제
      3. 가까운 미래에 양자컴퓨터가 현실화 될 수 있으며, 양자컴퓨터를 이용하여 공격할 경우 적어도 256 비트 크기의 키가 바람직

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - AES (Advanced Encryption Standard)
    - AES 공모의 모든 요구사항을 만족시킴
      - 128/192/256 bit keys, 128 bit data
      - an iterative rather than feistel cipher
      - 설계:
        - 하드웨어나 소프트웨어로 구현할 때 속도나 코드 간결성 (Compactness) 면에서 효율적
        - 알려진 블록 암호 알고리즘에 대한 공격들에 안전
        - 현재 AES에 대한 가장 실질적인 공격은 전수 키 조사
        - 최악의 경우(in the worst case) 2128 번의 계산이 필요 (이러한 계산량은 현재 가장 빠른 슈퍼컴퓨터가 계산을 수행해도 태양계의 수명보다 긴 시간이 필요)

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - AES (Advanced Encryption Standard)
    - 한 블록 : 128 비트
    - 128, 192, 256비트의 비밀키에 대해 라운드의 수는 각각 10, 12, 14라운드가 실행



Key size	n
128	10
192	12
256	14

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - AES (Advanced Encryption Standard)
    - DES와는 다르게 AES는 SW 효율적으로 구현되도록 설계
      - 바이트 단위의 연산을 주로 수행
      - 스마트카드와 같은 8 비트 프로세서에 효율적으로 설계
      - 하지만, 현재 PC에 주로 사용되는 32 비트나 64 비트 프로세서에서는 비효율적
        - Rijndael의 설계자들은 효율적인 SW 구현방법 제시
        - 한 라운드의 입력 값에 대응되는 출력 값을 표를 통해서 찾으면 되며, 모두 4개의 표가 존재
        - 표의 입력 값은 32 비트이며, 표를 특별히 T-Box라 부름
        - 1.2GHz 인텔 프로세서 사용시 초당 50MB 정도를 처리
      - AES는 DES보다 더 많은 하드웨어 자원을 요구
        - 집적 회로의 집적도(Integration Density)가 매우 높아짐. 현재 상용 AES ASIC의 경우 초당 10Ghz이상의 처리능력

CONTENTS

- 기타 대칭키 암호화 알고리즘

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 국내 대칭 암호화 방식
    - SEED 알고리즘
      - 전자상거래, 금융, 무선통신 등에서 전송되는 중요한 정보를 보호하기 위해 한국인터넷진흥원과 국내 암호 전문가들이 순수 국내 기술로 개발한 128비트 블록의 암호화 알고리즘
      - SEED 128은 1999년 9월 정보통신단체표준(TTA 표준)으로 제정되었고, 2005년에는 ISO/IEC와 IETF로부터 암호화 표준 알고리즘으로 인정
      - 국내에서 개발된 많은 암호 프로그램과 보안 솔루션에서 사용
    - ARIA 알고리즘
      - 전자정부 구현으로 다양한 환경에 적합한 암호화 알고리즘이 필요하여 국가보안기술연구소(NSRI) 주도로 개발
      - 2004년 국가표준기본법에 의거하여 국가표준Ⓚ으로 지정
      - AES 알고리즘과 마찬가지로 128/192/256비트 암호화 키 지원

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 기타 대칭 암호화 알고리즘
    - IDEA 알고리즘
      - 128비트 키를 사용하여 64비트 평문을 8라운드를 거쳐 64비트 암호문으로 만드는 방식
      - 모든 연산이 16비트 단위로 이루어지도록 하여 16비트 프로세서에서 구현이 용이하며 주로 키 교환에 쓰임
    - RC5 알고리즘
      - 비교적 간단한 연산으로 빠른 암호화와 복호화 기능을 제공하며 모든 하드웨어에 적합함
      - 입출력, 키, 라운드 수가 가변인 블록 알고리즘 RC5는 32, 64 또는 128비트 키가 사용되며 속도는 DES의 약 10배

대칭키 암호화 방식

- 대칭키 암호화(Symmetric key)
  - 기타 대칭 암호화 알고리즘
    - Skipjack 알고리즘
      - 미국 국가안보국(NSA)에서 개발한 클리퍼 칩에 내장된 블록 알고리즘
      - 비밀로 유지되던 알고리즘의 형태와 구조가 1998년에 공개
      - 64비트 입출력, 80비트 키를 이용하여 총 32라운드의 암호화 과정을 수행
    - LEA 알고리즘
      - 고속 환경 및 모바일 기기 등의 경량 환경에서 기밀성을 제공하기 위해 국내에서 개발된 암호화 알고리즘
      - AES 대비 1.5~2배의 성능임.
      - 128비트 데이터 블록으로 128, 192, 256비트 비밀 키를 사용할 수 있음

정보보호의 이해

- 대칭키 암호화 방식
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES) 소개

참고문헌

- 정보 보안 개론 - 한권으로 배우는 핵심 보안 이론, 양대일, 한빛 아카데미
- 현대 암호학 개론, 이동훈, 이룬 출판

Q & A