2026/05/14, 2026 보안 기초 세미나

암호학과 네트워크 보안

- 6장 DES, 7장 AES -

김 민 식 (ms6127@naver.com)

세종대학교 프로토콜공학연구실

목 차

- 보충
- DES
  - 구조
  - 보안성 및 취약성
  - 다중DES
- AES
  - 데이터단위
  - 구조
  - 보안성 및 안전성

보충

- 블럭암호 운영 모드간 비교

운영모드	패딩 필요	*IV*사용	블록 간 관계	처리 특성	스트림 암호 동작 여부	스트림 암호 종류
ECB	O	X	독립적	암호화 및 복호화 모두 병렬 처리	X	-
CBC	O	O	이전 암호문 의존	암호화 순차 처리 / 복호화 병렬 가능	X	-
CFB	X	O	이전 암호문 의존	암호화 순차 처리 / 복호화 병렬 가능	O	비동기식
OFB	X	O	이전 출력 값 의존	암호화 순차 처리 / 복호화 순차처리	O	동기식
CTR	X	O	독립적	암호화 및 복호화 모두 병렬 처리	O	동기식

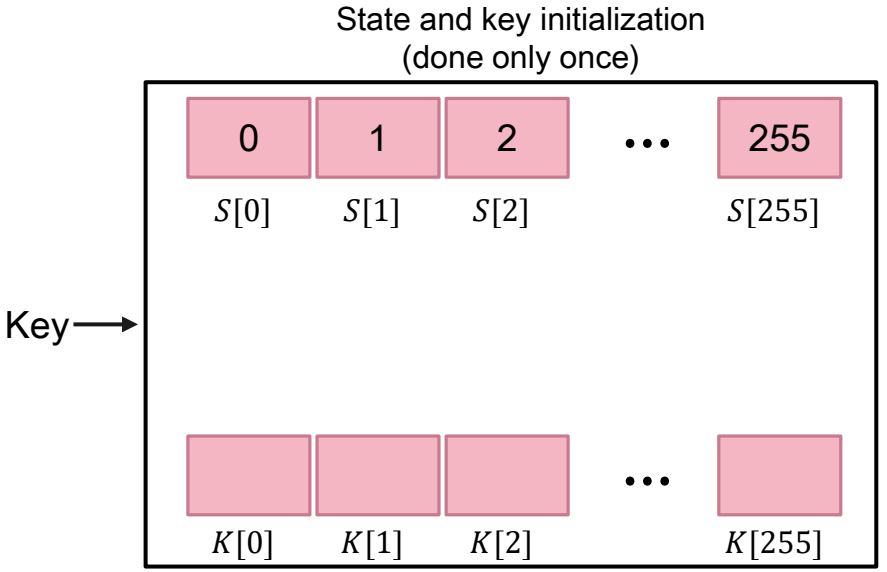
보충

- 스트림 암호(Stream Cipher)
  - 정의
    - 평문 스트림과 키 스트림을 XOR하여 암호문 스트림을 생성하는 암호 방식
  - 특징
    - 비트 또는 바이트 단위의 데이터를 암호화하기 위해 사용함
    - 키 스트림을 생성한 뒤 평문과 XOR하여 암호화함
    - 데이터를 블록으로 모아서 처리하지 않고, 연속적으로 바로 암호화할 수 있기 때문에 실시간 처리에 유리함
  - 종류
    - RC4
    - A5/1

보충

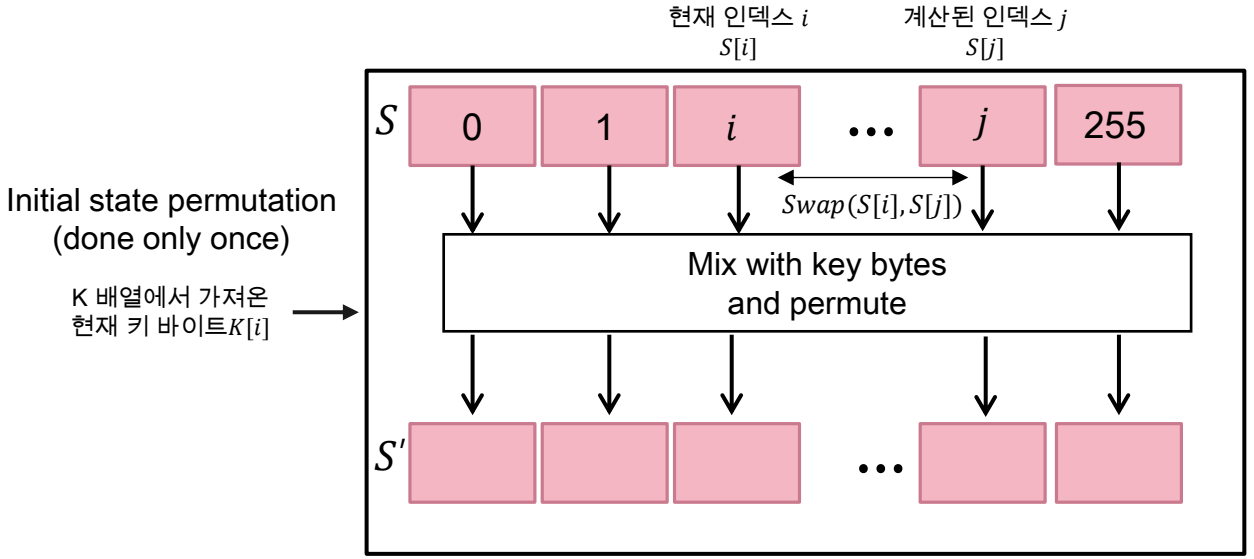
- RC4 (1/6)
  - 정의
    - 256바이트 상태 배열을 기반으로 키 스트림을 생성하고 이를 평문과 XOR하여 암호화하는 스트림 암호
  - 상태(state)
    - 256바이트로 구성된 상태 배열 $S[0], S[1], \ldots, S[255]$을 사용함
    - 각 원소는 0부터 255 사이의 1바이트 값을 가짐
    - 상태 배열을 계속 치환하며 키 스트림을 생성함

- RC4 (2/6)
  - 동작과정
    1. 상태 배열 S 초기화
       - $S[i]$ 는 0부터 255까지 순차적으로 초기화됨
       - $K[i]$ 는 256바이트가 될 때까지 비밀 키를 반복 저장함




```
For (i = 0 to 255)
{
    S[i] ← i
    K[i] ←key [i mod key Length]
}
```

- RC4 (3/6)
  - 동작과정
    2. KSA(Key Scheduling Algorithm)를 통한 초기 치환
       - $j = 0$ 으로 초기화함
       - $i = 0$ 부터 255까지 반복하며, $j = (j + S[i] + K[i])\ mod\ 256$ 계산
         - i = 현재 차례대로 보고 있는 위치
           j = 키와 $S[i]$ 값을 이용해 새로 계산되는 교환 위치
       - $S[i]$와 $S[j]$를 교환하여 상태 배열을 섞음



$j \leftarrow 0$
for$(i = 0\ to\ 255)$
{
  $j \leftarrow (j + S[i] + K[i])mod\ 256$
  $swap(S[i], S[j])$
}

비밀 키 $K[i]$를 이용해 S 배열을 섞는 과정임

보충

- RC4 (4/6)
  - 동작과정

3. 키 스트림 생성
   - i 와 j 값을 갱신함
   - 갱신된 i, j 위치의 $S[i]$와 $S[j]$를 교환함
   - $S[(S[i] + S[j])mod256]$ 값을 키 스트림 바이트 k로 출력함

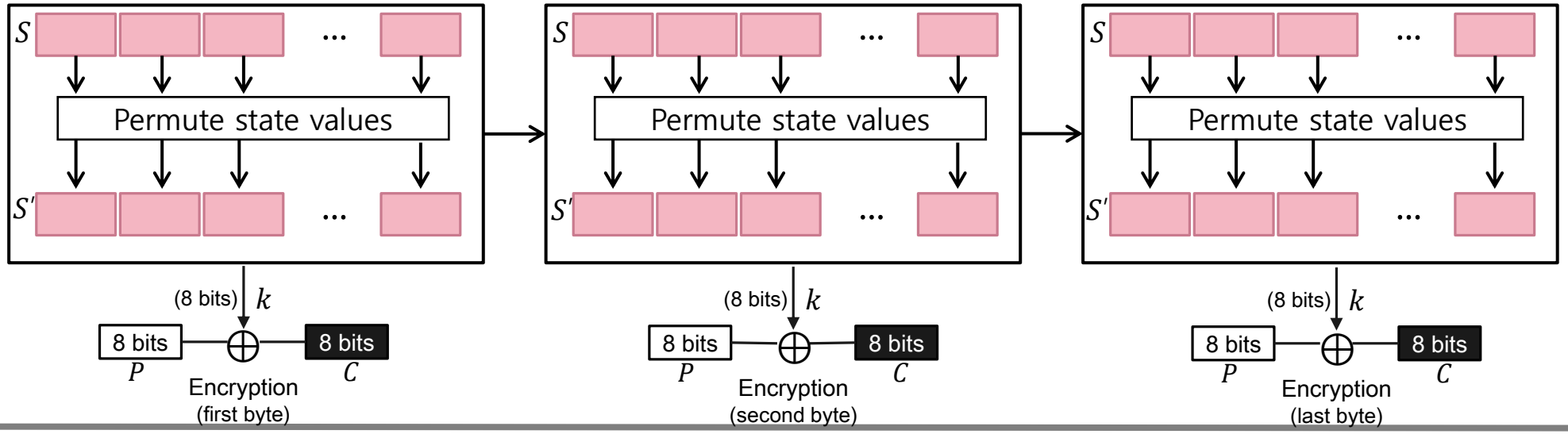
$i \leftarrow (i + 1)\ mod\ 256$
$j \leftarrow (j + S[i])\ mod\ 256$
$Swap\ (S[i], S[j])$
$k \leftarrow S[(S[i] + S[j])mod\ 256$

KSA로 섞인 S 배열을 이용해 i, j를 갱신하고
갱신된 S배열에서 k를 출력

- RC4 (5/6)
  - 동작과정
    4. 암호화 및 복호화
       - 암호화: 키 스트림 k와 평문 P를 XOR하여 암호문 C 생성

         $$C = P \oplus k$$
       - 복호화: 같은 키 스트림 k와 암호문 C를 XOR하여 평문 P 복원

         $$P = C \oplus k$$
       - XOR 연산의 성질 때문에 암호화와 복호화는 동일한 구조를 가짐



보충

- 현대 대칭-키 암호를 이용한 암호화 기법
  - 스트림 암호(Stream Cipher)
    - RC4 (6/6)
      - 보안성 문제
        - KSA의 구조적 취약점
          - RC4는 KSA 과정에서 비밀 키를 이용해 상태 배열 S를 섞음
          - 섞는 과정이 완전한 무작위가 아니어서 초기 S배열과 키 스트림에 편향이 남을 수 있음
          - 짧은 키(40/56/64비트)는 전수조사 공격에 취약함
          - 키 길이를 늘려도 섞는 방식이 완전히 무작위가 아니어서 초기 S 배열과 키 스트림에 일정한 패턴이 남아 안전성을 보장할 수 없음
        - 통계적 분석 공격 가능
          - 초기 키 스트림의 편향을 이용해 여러 암호문을 비교, 분석하면 키 바이트나 평문 정보를 추정할 수 있음
          - 동일 키 재사용 시 같은 키 스트림이 반복되어 보안성이 더욱 약화됨

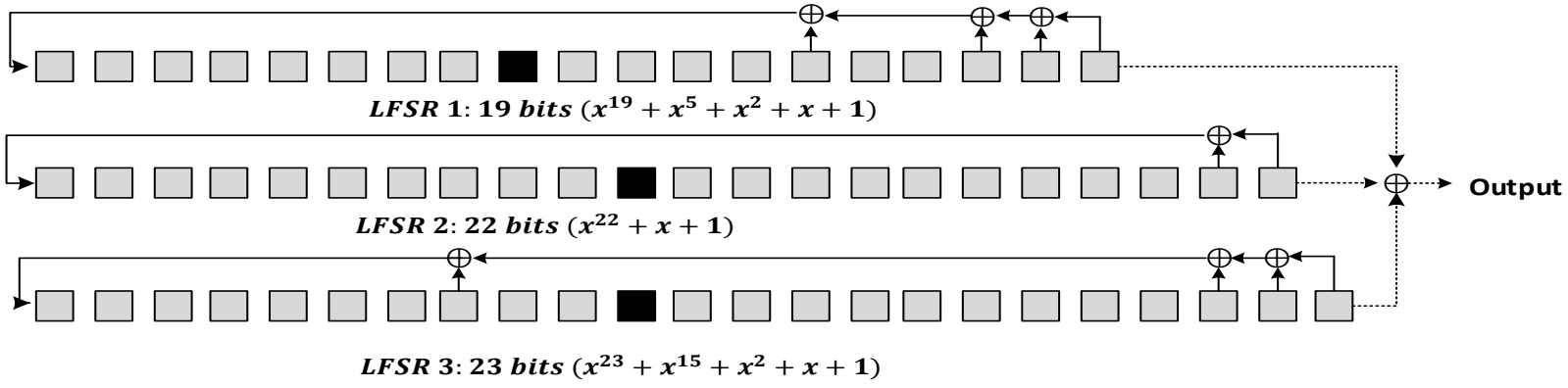
보충

- A5/1 (1/5)
  - 정의
    - LFSR 기반으로 키 스트림을 생성하는 스트림 암호 알고리즘임
  - 특징
    - 3개의 LFSR을 이용해 키 스트림을 생성함
    - 다수결 클록 제어 방식을 사용해 일부 레지스터만 불규칙적으로 이동시킴
    - 블록 단위가 아닌 비트 단위로 동작하는 동기식 스트림 암호임

보충

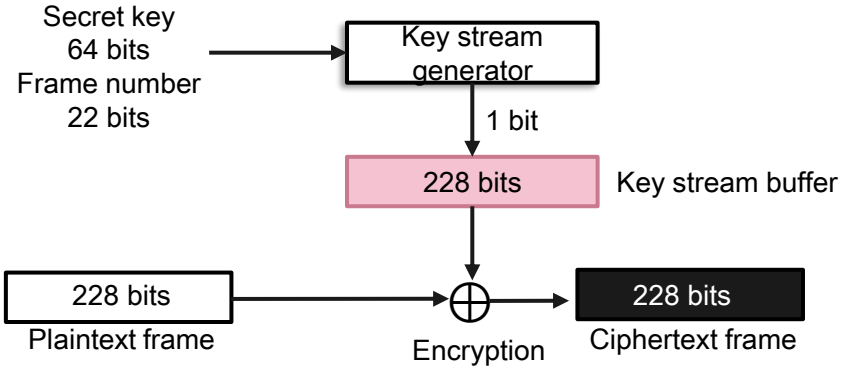
- A5/1 (2/5)
  - 키 생성기
    - 길이가 19, 22, 23인 3개의 LFSR을 사용함
    - 각 LFSR은 고유한 특성 다항식을 가지며, 클로킹 비트를 기준으로 이동 여부가 결정됨
    - 세 LFSR의 출력이 결합되어 최종 키 스트림 비트를 생성함
    - 생성된 키 스트림은 228비트 버퍼에 저장되어 평문 프레임과 XOR됨
      - 보통 GSM에서 한 프레임을 암호화할 때 보통 114비트씩 두 방향에 사용하기 때문임

*클로킹
LFSR의 비트들을 한 단계 이동시키고 상태를 갱신하는 동작



보충

- A5/1 (3/5)
  - 동작과정 (1/3)
    1. 초기화
       - 세 LFSR의 모든 셀을 0으로 설정한 뒤, 64비트 비밀 키와 22비트 프레임 번호를 이용해 내부 상태를 구성함
         - 현재 통신 프레임을 구분하기 위한 값이며, 비밀 키가 아니라 프레임마다 달라지는 입력값임
       - 프레임 번호는 각 통신 프레임을 구분하기 위한 22비트 값으로, 같은 비밀 키를 사용하더라도 프레임마다 다른 키 스트림이 생성되도록 함
       - 이후 22비트 프레임 번호를 이용해 100회 클로킹하여 내부 상태를 충분히 섞은 뒤, 키 스트림을 생성함



보충

- A5/1 (4/5)
  - 동작과정 (2/3)
    2. 키 스트림 생성
       - 매 단계마다 먼저 majority 함수를 계산함
         - 세 개의 클로킹 비트 중 둘 이상이 1이면 1, 아니면 0을 출력함
       - majority 값과 일치하는 LFSR만 클로킹함
       - 클로킹 후 각 LFSR의 출력 비트를 XOR하여 1비트 키 스트림을 생성함
       - 이 과정을 반복하여 전체 키 스트림을 만듦

보충

- A5/1 (5/5)
  - 동작과정 (3/3)
    3. 암호화
       - 생성된 키 스트림은 228비트 버퍼에 저장됨
       - 평문 프레임과 키 스트림을 XOR하여 암호문 프레임을 생성함
    4. 복호화
       - 복호화도 같은 키 스트림과 XOR하여 수행함
       - 암호화와 복호화는 동일한 XOR 구조를 가짐
  - 보안성 문제
    - 64비트 세션 키는 현대 기준에서 짧아 안전성이 충분하지 않음
    - 평문 일부가 알려지면 암호문과 XOR하여 키 스트림 일부를 추정할 수 있음
    - 추정된 키 스트림과 LFSR 구조를 분석하면 내부 상태나 세션 키가 노출될 수 있음

목 차

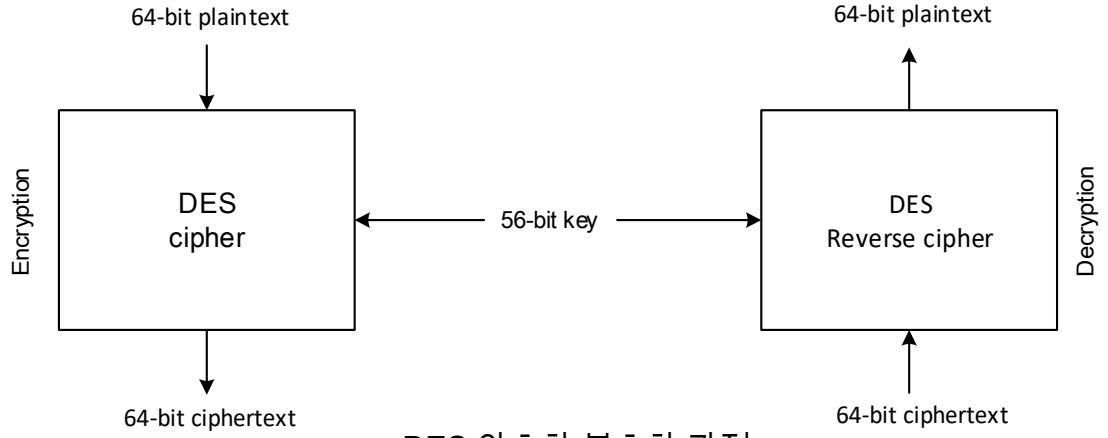
- 보충
- DES
  - 구조
  - 보안성 및 취약성
  - 다중DES
- AES
  - 데이터단위
  - 구조
  - 보안성 및 안전성

DES (Data Encryption Standard)

- 개요
  - 역사
    - 1973년
      - 미국 국립기술표준원이 국가적으로 사용할 대칭 키 암호시스템의 제안 요청서 발표
    - 1975년
      - IBM이 개발한 DES가 채택
    - 1977년
      - 연방관보의 FIPS 46이란 이름으로 발표

DES (Data Encryption Standard)

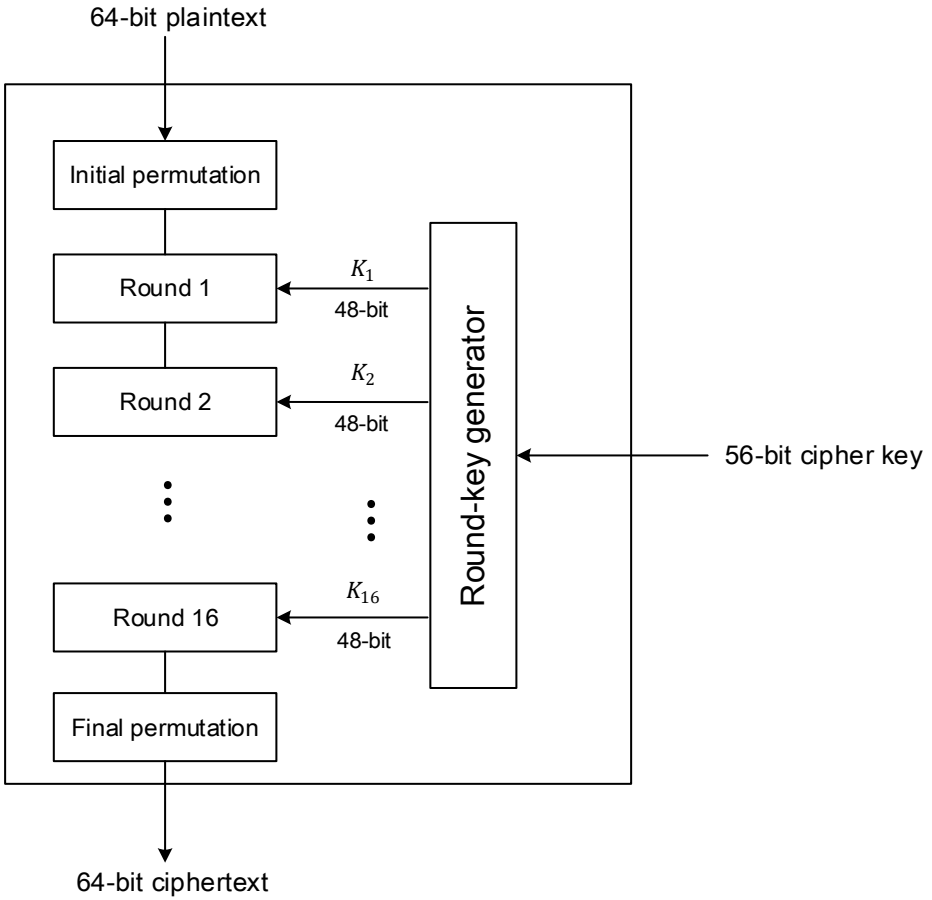
- 개요
  - 정의
    - 하나의 비밀키를 사용하여 데이터를 암호화하는 대칭 키 블록 암호 알고리즘임
  - 특징
    - 64비트 평문 블록을 64비트 암호문으로 변환함
    - 2개의 P-Box 전치와 16라운드 Feistel 구조를 사용함



DES 암호화 복호화 과정

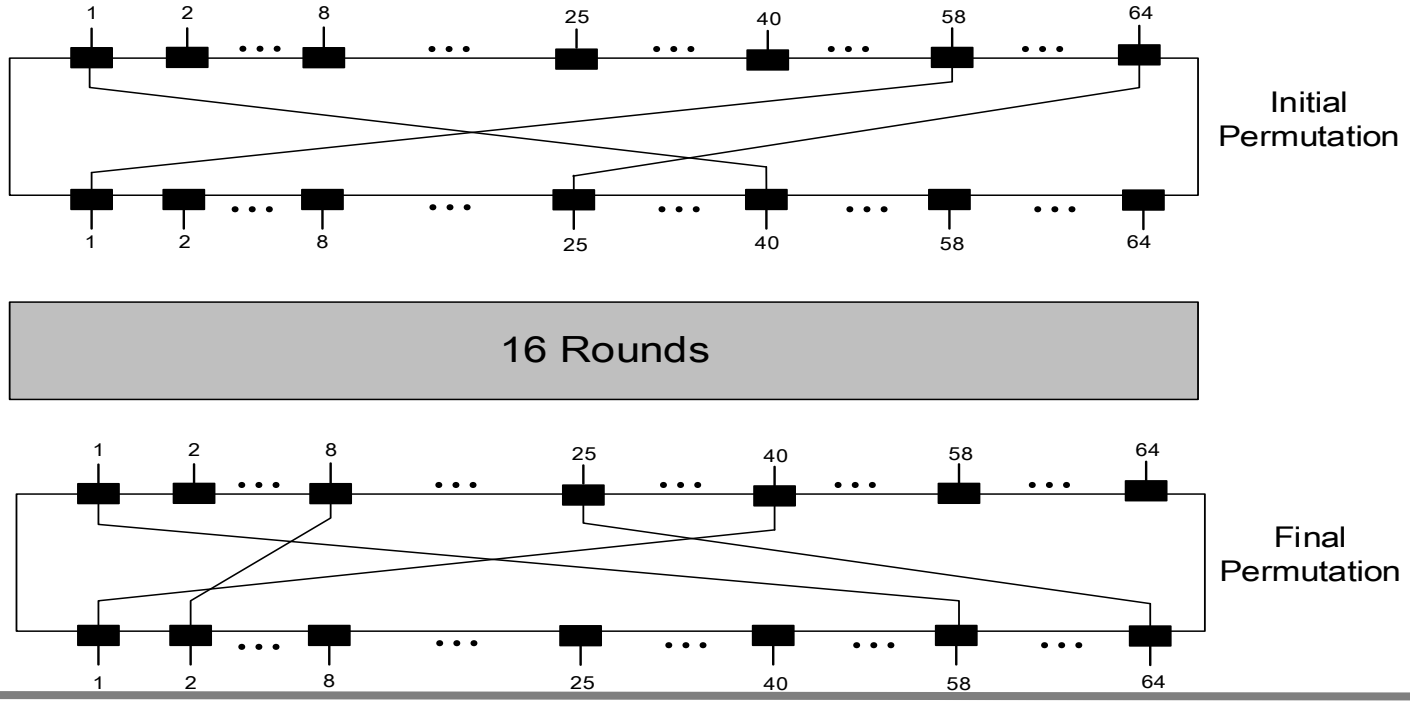
DES (Data Encryption Standard)

- 구조
  - 초기 치환(Initial permutation)/최종 치환(final permutation)이라고 부르는 두 개의 P-박스와 16개의 라운드로 구성



DES (Data Encryption Standard)

- 초기 치환과 최종 치환
  - 64비트 입력 블록의 비트 위치를 미리 정해진 규칙에 따라 재배열하는 과정임
  - 초기 치환은 16라운드 수행 전에 적용됨
  - 최종 치환은 16라운드 수행 후 적용되며, 초기 치환의 역순 구조임



DES (Data Encryption Standard)

- 초기 치환과 최종 치환
  - 치환표
    - 치환표의 값은 입력 비트의 위치를 의미함
      - e.g., 초기 치환표에서 첫 번째 값 58은 입력의 58번째 비트가 출력의 1번째 비트로 이동함을 의미함
    - 최종 치환은 초기 치환의 역치환 구조임

Initial Permutation(IP)							
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

Final Permutation(FP)							
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

IP에서 입력 58번째 비트가 출력 1번째 위치로 이동
FP에서 입력 1번째 비트가 출력 58번째 위치로 이동

DES (Data Encryption Standard)

- 초기 치환과 최종 치환
  - 예제 6.1

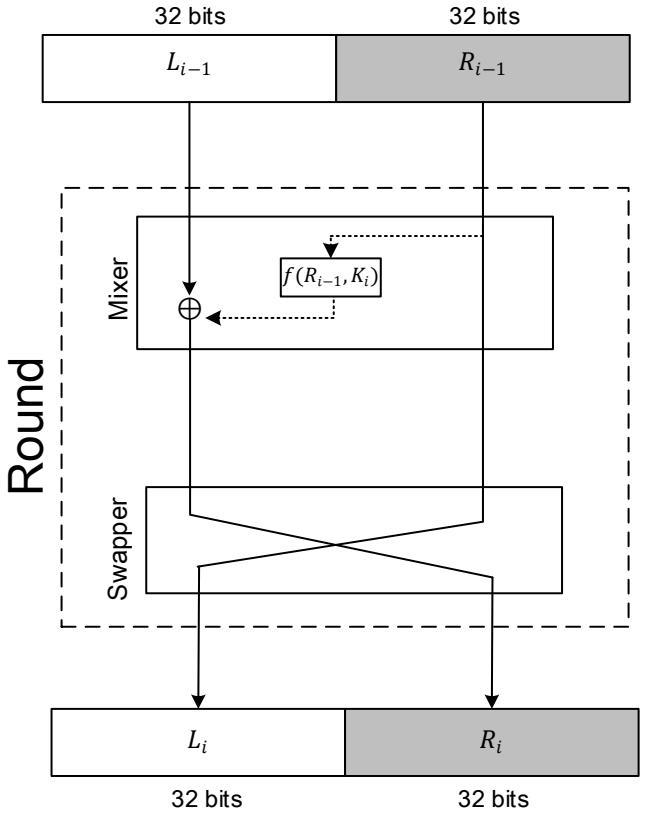
입력 값이 다음과 같은 16진수일 때, 초기 치환에 의한 출력 값을 구하시오.
0x0002 0000 0000 0001

- 입력 값에서 1인 비트: 15번째, 64번째
  - 16진수 입력 값을 2진수로 변환했을 때의 최상위 비트 위치로부터 숫자 1의 위치
    - 변환:00000000 00000010 00000000 00000001
- IP 표를 통해 위치 확인
  - 입력 64번째 비트 → 출력 25번째 비트
  - 입력 15번째 비트 → 출력 63번째 비트
- 출력 값
  - 0x0000 0080 0000 0002

Initial Permutation(IP)							
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

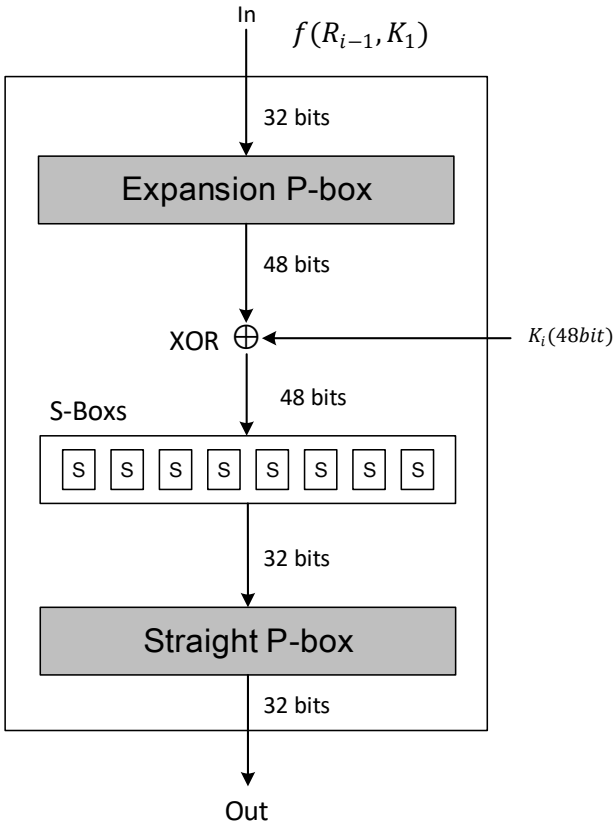
DES (Data Encryption Standard)

- 라운드(Round)
  - DES는 총 16번의 라운드를 반복하여 암호화를 수행함
  - 라운드 과정에서는 Mixer와 Swapper가 활용되며, 이때 Mixer에서 활용되는 라운드 함수 f는 DES 함수임



DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수
    - 구성
      - 확장 P-박스, XOR 연산,라운드 키, 8개의 S-박스, 단순 P-박스
    - 단계
      1. 확장 P-박스를 통해 32비트 입력을 48비트로 확장함
      2. 라운드 키 K_i 와 48비트 블록을 XOR 연산함
      3. 8개의 S-Box를 거쳐 48비트 블록을 32비트로 축약함
      4. 단순 P-박스를 거쳐 32비트 블록을 전치함

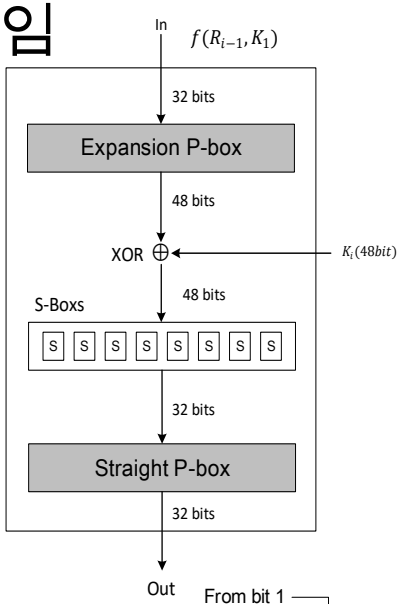


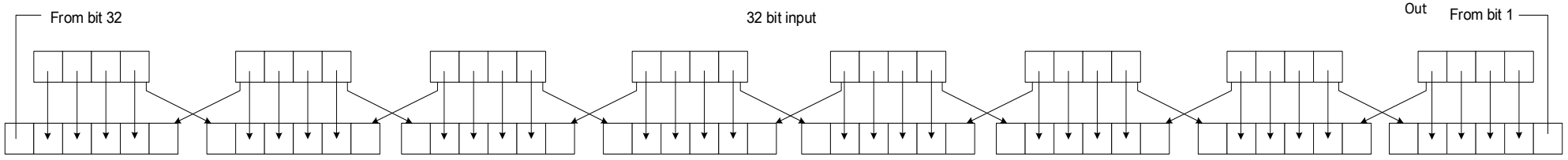
DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수
    - (1단계)확장 P-박스 (E table, Expansion/Permutation)
      - R_{i-1} 의 32비트를 48비트로 확장함
      - 라운드 키 K_i와 XOR하기 위해 비트 수를 맞추는 과정임
      - 8개의 4비트 블록을 각각 6비트로 확장함
      - 일부 인접 비트가 중복되어 확장됨

DES (Data Encryption Standard)

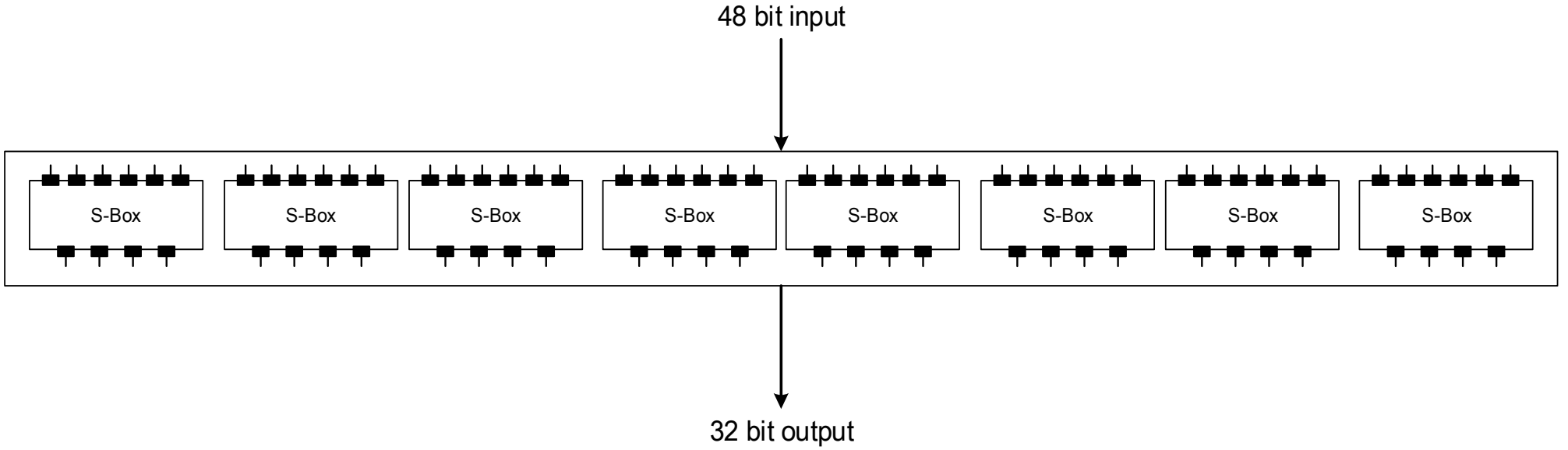
- 라운드(Round)
  - DES 함수
    - (2단계) 라운드 키와 XOR연산
      - 확장 P-박스를 거친 48비트 블록과 라운드 키 K_i를 XOR 연산함
      - 라운드 키 K_i는 키 생성 과정에서 만들어진 48비트 키임
      - XOR 결과는 다음 단계인 S-Box의 입력으로 사용됨





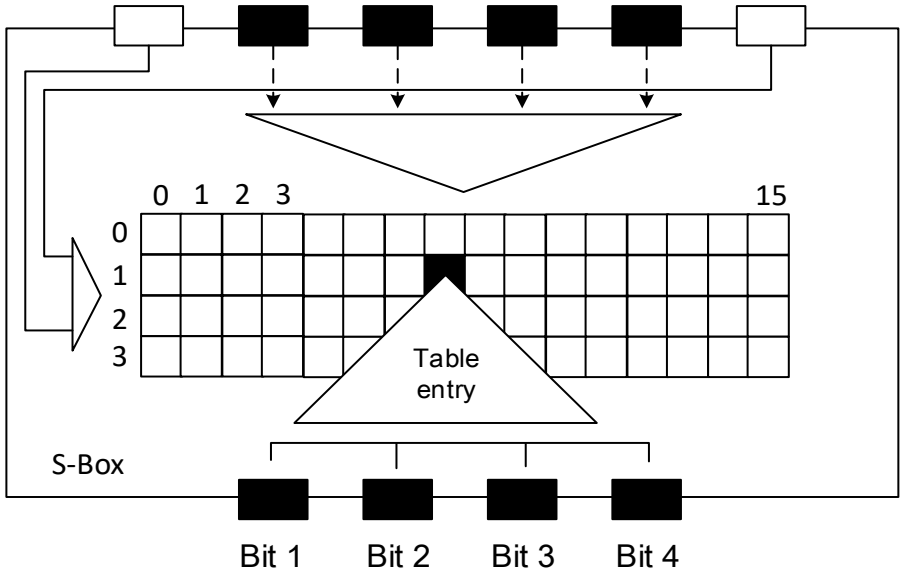
DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수
    - (3단계) S-Box 대치
      - 48비트 입력을 6비트씩 8개로 나누어서,가각 8개의 S-Box에 입력됨
      - 각 S-Box는 6비트 입력을 4비트 출력으로 변환함
      - 결과적으로 48비트 값이 32비트 값을 생성함



DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수
    - (3단계) S-Box 대치
      - S-Box 표 참조 과정
        - 6비트 입력의 첫 번째와 마지막 비트는 S-Box 표에서 참조할 행의 순번을 결정함
        - 가운데 4개의 비트는 S-Box 표에서 참조할 열의 순번을 결정함
        - 해당 행과 열이 만나는 위치의 값이 출력 값이 됨
        - 출력 값은 4비트로 표현됨



DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수 - 예제 6.3

8개의 S-Box 중, 첫 번째 S-Box의 입력 값이 100011이다. 출력 값은 무엇인가?

- 첫 번째와 여섯 번째 비트를 합치면 2진수로 11이며, 10진수로 3
- 남아있는 비트는 2진수로 0001이며, 10진수로 1
- S-Box 3행 1열의 값을 구하면, 그 결과 10진수로 12이고 2진수로 1100

	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

DES (Data Encryption Standard)

- 라운드(Round)
  - DES 함수
    - (4단계) 단순 P-박스 전치
      - S-Box를 거친 32비트 값을 단순 P-박스에 입력함
      - 비트 값은 바꾸지 않고 위치만 재배열함
      - 재배열된 32비트 값은 DES 함수의 최종 출력 값이 됨
      - 다음 라운드 연산에서 확산 효과를 높이는 역할을 함

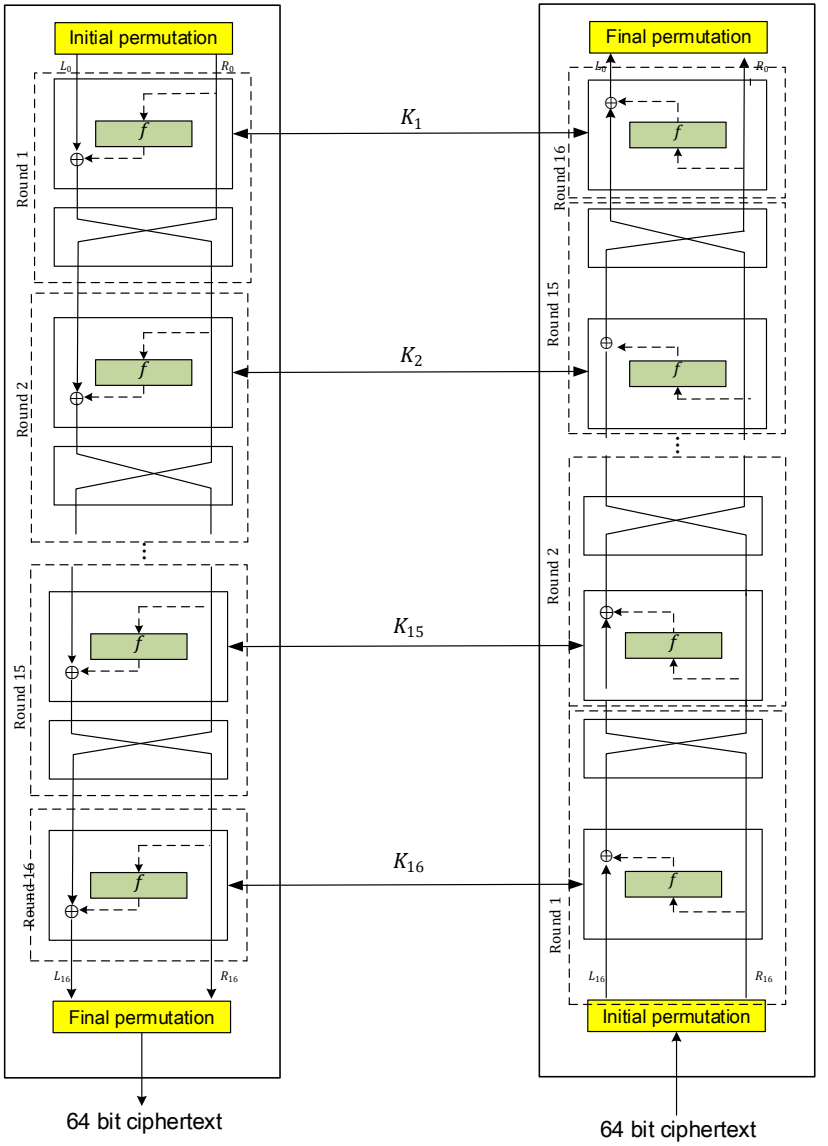
DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 구성요소
    - 두 개의 P-Box(초기 치환, 최종 치환), 16 라운드의 feistel 암호
  - 특징
    - 암호 알고리즘과 복호 알고리즘은 서로 역함수 관계
    - 복호화 과정에서는 암호화와 동일한 feistel 라운드 구조를 사용함

DES (Data Encryption Standard)

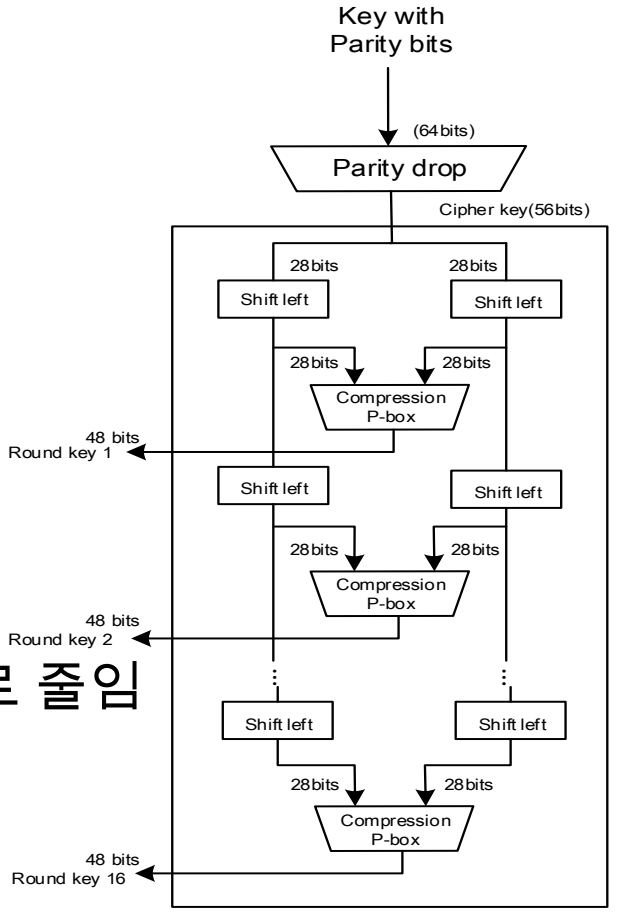
- 암호화와 복호화 알고리즘
  - 라운드 반복 구조와 초기 치환 및 최종 치환의 역 관계를 이용함
  - 복호화는 라운드 키 K_i의 사용 순서를 반대로 적용함
    - 암호화: $K_1 \rightarrow K_2 \rightarrow \ldots \rightarrow K_{16}$
    - 복호화: $K_{16} \rightarrow K_{15} \rightarrow \ldots \rightarrow K_1$
  - 일반화 식
    - $L_i = R_{i-1}$
    - $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

      (단, 마지막 라운드 제외)



DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 라운드 키 생성
    - 64비트 암호 키로부터 16개의 48비트 라운드 키를 생성함
  - 단계
    1. 패리티 비트 제거
       - 64비트 키에서 8개의 패리티 비트를 제거하여 56비트 키를 생성함
    2. 왼쪽 순환 이동
       - 56비트 키를 28비트씩 두 블록으로 나눔
       - 각 라운드마다 정해진 비트 수만큼 왼쪽으로 순환 이동함
    3. 축소 치환
       - 56비트 값을 축소 P-Box에 적용하여 48비트로 줄임
    4. 라운드 키 생성
       - 위 과정을 16번 반복하여 $k_1, \ldots, k_{16}$을 생성함



DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 라운드 키 생성
    - (1단계) 패리티 비트 제거
      - 64비트 암호 키에서 8(8, 16, 24, ...,64)개의 패리티 비트를 제거함
      - 실제 라운드 키 값으로 사용되는 56비트 키를 생성함
      - 패리티 비트 제거 표에 따라 비트 위치를 재배열함
      - 생성된 56비트 키는 다음 단계에서 두 개의 28비트 블록으로 나누어짐

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

패리티 비트 제거 표

DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 라운드 키 생성
    - (2단계) 좌측 순환 이동
      - 패리티 비트 제거 후 생성된 56비트 키를 28비트씩 두 블록으로 나눔
      - 각 블록을 라운드에 따라 왼쪽으로 1비트 또는 2비트 이동함
      - 이동된 비트는 사라지지 않고 반대쪽 끝으로 이동함
      - 이동 결과는 다음 단계인 축소 P-Box의 입력으로 사용됨

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 라운드 키 생성
    - (3단계) 축소 치환
      - 좌측 순환 이동이 끝난 두 28비트 블록을 다시 합쳐 56비트 값으로 만듦
      - 축소 P-Box를 활용하여 56비트 값을 48비트로 축약함
      - 생성된 48비트 값이 해당 라운드의 라운드 키 K_i가 됨

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

축소 P-Box 표

DES (Data Encryption Standard)

- 암호화와 복호화 알고리즘
  - 라운드 키 생성
    - (4단계) 반복 수행
      - 좌측 순환 이동과 축소 치환 과정을 반복함
      - 각 라운드마다 새로운 48비트 라운드 키가 생성됨
      - 총 16개의 라운드 키 $K_1, \ldots, K_{16}$을 생성함

DES (Data Encryption Standard)

- 보안성 분석 요소
  - 쇄도 효과(avalanche effect)
  - 완전성 효과(completeness)

DES (Data Encryption Standard)

- 보안성 분석 요소
  - 쇄도 효과
    - 암호 알고리즘이 입력 값에 미세한 변화를 줄 경우 전체 출력으로 퍼지는 변화가 일어나는 성질임
    - DES는 라운드 반복을 거치며 작은 차이를 전체 출력으로 확산시킴

DES (Data Encryption Standard)

- 보안성 분석 요소
  - 쇄도 효과
    - 예제 6.2

DES에서 쇄도 효과를 검사하기 위하여, 한 비트만 다른 두 개의 평문 블록을 동일한 키를 가지고 암호화하고, 각 라운드의 비트들의 수에 있어서 차이점을 관찰하라

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

- 평문: 0000000000000000
  키: 22234512997ABB23
  암호문: 4789FD476E82A5F1

  평문: 0000000000000001
  키: 22234512997ABB23
  암호문: 0A4ED5C15A63FEA3

- 2개의 평문 블록이 가장 오른쪽의 비트만 다를지라도, 암호문은 29 비트가 다름. 평문의 약 1.5%가 변하는 것은 암호문의 약 45%의 변화를 나타낼 수 있음을 의미

DES (Data Encryption Standard)

- 보안성 분석 요소
  - 완전성 효과
    - 암호문의 각 비트가 평문과 키의 여러 비트에 영향을 받아 결정되는 성질임
    - DES에서는 S-Box와 P-Box를 통해 입력과 키의 영향이 전체 출력으로 퍼지게 하여 확산성과 혼돈성을 높임

DES (Data Encryption Standard)

- 취약성
  - 알고리즘 설계 취약성
    - S-Box 설계 문제점
    - P-Box 설계 문제점
    - 라운드 수 문제점
  - 암호키 취약성
    - 암호 키 길이 문제점
    - 취약 키 문제점
    - 준 취약 키 문제점
    - 잠재적 취약 키 문제점
    - 키 보수 문제점

DES (Data Encryption Standard)

- 알고리즘 설계 취약성
  - S-Box 설계 문제점
    - 일부 S-Box에서 입력과 출력 사이의 특정 관계가 존재함
    - 입력 값을 바꿨을 때 출력 변화가 완전히 무작위적이지 않고, 특정 패턴을 보일 가능성이 있음
    - 특정 입력 차이가 특정 출력 차이로 나타나는 경향이 있어 차분 분석에 이용될 수 있음

DES (Data Encryption Standard)

- 알고리즘 설계 취약성
  - P-Box 설계 문제점
    - 초기 치환과 최종 치환을 사용한 이유가 명확하지 않음
    - 초기 치환과 최종 치환은 암호학적 강도에 직접적인 영향을 주지 않음
    - 확장 P-Box에서는 일부 비트가 반복되어 사용됨

DES (Data Encryption Standard)

- 암호 키 취약성
  - 암호 키 길이
    - DES는 실제 암호 연산에 56비트 키를 사용함
    - 2^{56}개의 키 공간은 과거에는 충분했지만,
      현재 기준에서는 전수조사 공격이 가능한 수준임
      - 현대 대칭키 암호에서는 최소 112비트 이상, 장기적 안전성을 위해 128비트 이상의 보안 강도가 권장됨

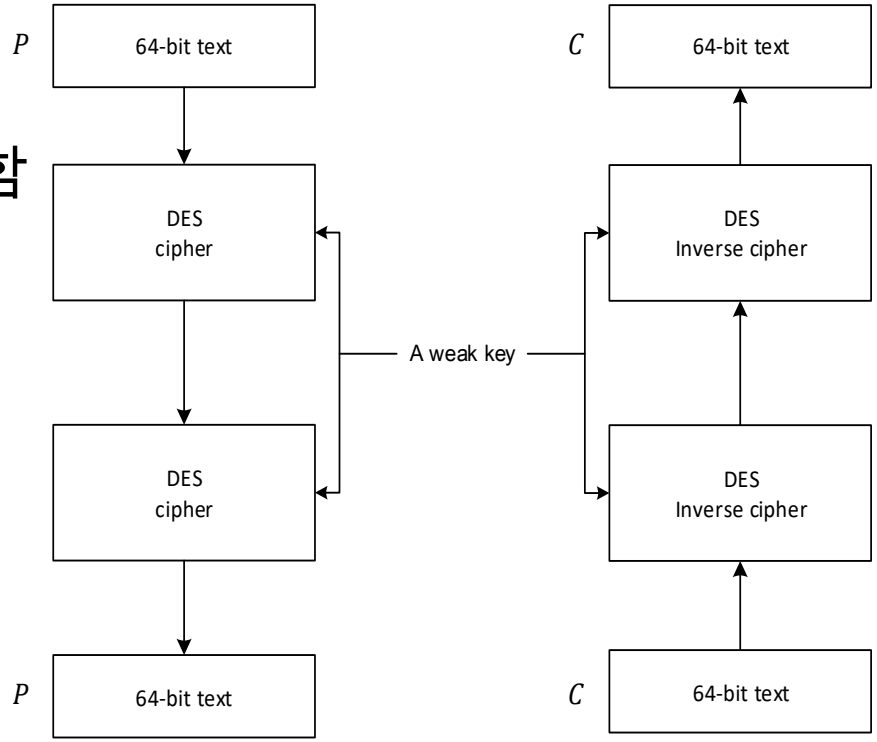
DES (Data Encryption Standard)

- 암호 키 취약성
  - 취약 키(Weak Keys)
    - 정의
      - 패리티 비트 제거 이후 특정한 반복 패턴을 갖는 키임
    - 특징
      - 모두 0, 모두 1, 또는 절반은 0이고 절반은 1인 형태로 구성됨
      - 취약 키로 생성된 라운드 키들은 서로 동일한 패턴을 가짐
      - 라운드 키가 반복되어 DES의 라운드 구조가 단순해짐, 이 경우 공격자는 취약 키 후보를 빠르게 검사할 수 있다는 문제점이 발생함

64비트 키(패리티 비트 포함 키)	56비트 키(실제 키)
0101 0101 0101 0101	00000000 00000000
1F1F 1F1F 1F1F 1F1F	00000000 FFFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFFF 00000000
FEFE FEFE FEFE FEFE	FFFFFFFF FFFFFFFF

DES (Data Encryption Standard)

- 암호 키 취약성
  - 취약 키(Weak Keys)
    - 문제점
      - 동일한 취약 키로 두 번 암호화하면 원래 평문으로 돌아올 수 있음
      - 암호화와 복호화가 대칭적인 형태를 가짐
        - $E_K\left(E_K(P)\right) = P$
      - 공격자는 취약 키 후보를 빠르게 검사할 수 있다는 문제점이 발생함



DES (Data Encryption Standard)

- 암호 키 취약성
  - 준 취약 키(Semi Weak Keys)
    - 정의
      - 서로 쌍을 이루는 두 개의 DES 입력 키를 의미함
    - 특징
      - 두 키가 하나의 키 쌍(pair) 을 이룸
      - K_A, K_B는 라운드 키가 아니라 DES에 입력되는 암호 키임
      - 두 입력 키에서 생성되는 라운드 키들이 서로 역순 관계를 가짐
      - DES의 라운드 구조가 단순해져 안전성이 약화될 수 있음
    - $E_{K_B}\left(E_{K_A}(P)\right) = P$

First key in the pair	Second key in the pair
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E02F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

DES (Data Encryption Standard)

- 암호 키 취약성
  - 준 취약 키(Semi Weak Keys)
    - 대응 관계
      - 한 쌍으로 묶인 두 키는 동일한 라운드 키들을 생성함
      - 차이점은 라운드 키의 생성 순서가 서로 반대라는 점임
      - 키 A의 라운드 키 순서가 K1 → K2 → ... → K16이라면, 키 B는 K16 → K15 → ... → K1 형태로 대응됨
      - 두 키는 암호화와 복호화가 서로 대응되는 관계를 가짐

DES (Data Encryption Standard)

- 암호 키 취약성
  - 잠재적 취약 키(Possible Weak Keys)
    - 정의
      - DES 입력 키 중에서 16개의 라운드 키가 모두 다르게 생성되지 않고, 일부 라운드 키가 반복되는 키임
    - 특징
      - 일부 라운드 키가 반복되어 라운드별 키 다양성이 약해짐
      - 취약 키나 준 취약 키처럼 바로 평문이 복원되는 것은 아니지만, DES 구조를 단순하게 만들 수 있어 공격자가 키 패턴을 이용해 분석할 가능성이 생김
        - eg., 1F1F01010E0E0101, FEFE0101FEFE0101
          - 1F, 1F, 01, 01, 0E, 0E, 01, 01 처럼 반복 배턴을 가짐
          - 키 스케줄에서 순환 이동을 해도 새로운 라운드 키가 충분히 생성되지 않음

DES (Data Encryption Standard)

- 암호 키 취약성
  - 키 보수
    - 정의
      - 키와 평문을 각각 비트 반전하면, 암호문도 비트 반전된 형태로 나타나는 성질임
    - 특징
      - 원래 키와 보수 키 사이에 일정한 암호문 관계가 존재함
      - 암호문의 무작위성을 약화시키고 키 탐색을 일부 줄일 수 있음
      - $C = E(K, P) \rightarrow = \overline{C} = \mathrm{E}(\overline{K}, \overline{P})$

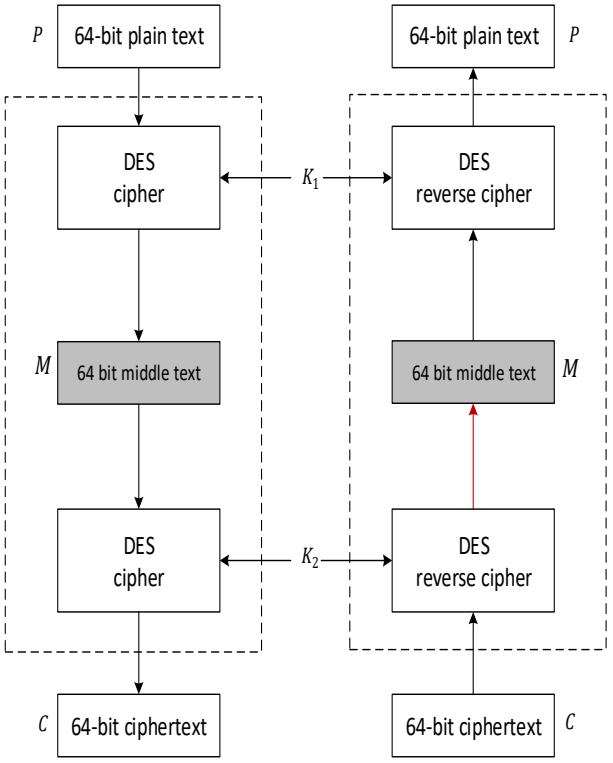
Notation	Original	Notation	Complement
K	1234123412341234	$\overline{K}$	EDCBEDCBEDCBEDCB
P	12345678ABCDEF12	$\overline{P}$	EDCBA987543210ED
C	E112BE1DEFC7A367	$\overline{C}$	1EED41E210385C98

DES (Data Encryption Standard)

- 다중 DES
  - 개요
    - 단일 DES의 짧은 키 길이를 보완하기 위해 등장한 방식임
    - 하나의 평문에 DES 알고리즘을 여러 번 적용함
    - 여러 개의 키를 사용하여 전수조사 공격의 난이도를 높임
  - 종류
    - 이중 DES(Double DES)
      - 두 개의 서로 다른 키로 DES를 두 번 적용함
    - 삼중 DES(Triple DES)
      - 두 개 또는 세 개의 키로 DES를 세 번 적용함

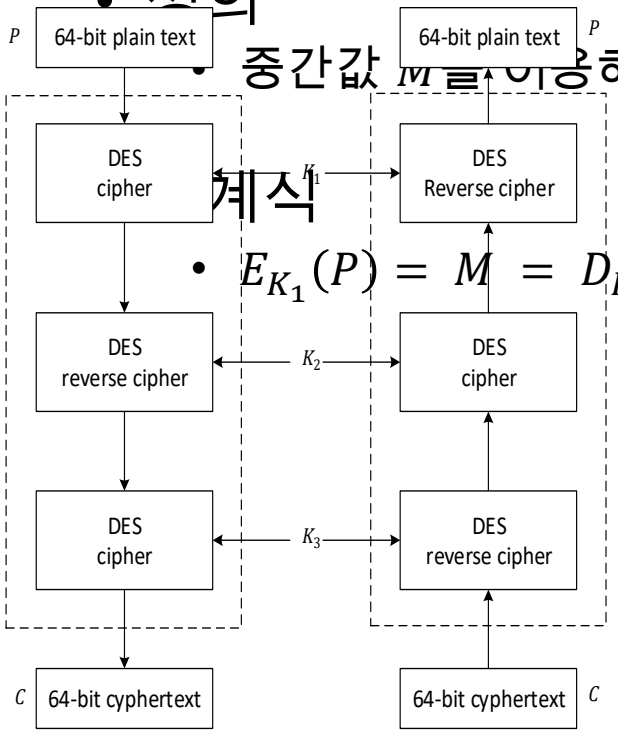
DES (Data Encryption Standard)

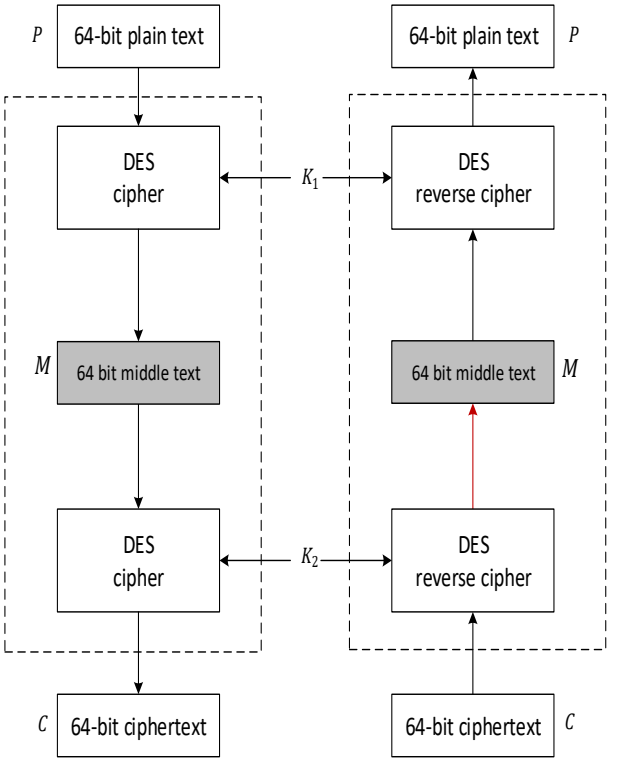
- 다중 DES
  - 이중 DES(2DES, Double DES)
    - 정의
      - DES 암호화 과정을 두 번 반복하는 방식임
      - 서로 다른 두 개의 키 K_1, K_2를 사용함
    - 알고리즘
      - $M = E_{K_1}(P)$
      - $C = E_{K_2}(M)$
    - 특징
      - 단일 DES보다 키 공간이 커진 것처럼 보임
      - 중간 일치 공격에 취약함



DES (Data Encryption Standard)

- 다중 DES
  - 이중 DES(2DES, Double DES)
    - 중간 일치 공격
      - 정의
        - 중간값 M을 이용해 키를 찾는 공격 방식임
      - 계산
        - $E_{K_1}(P) = M = D_{K_2}(C)$





DES (Data Encryption Standard)

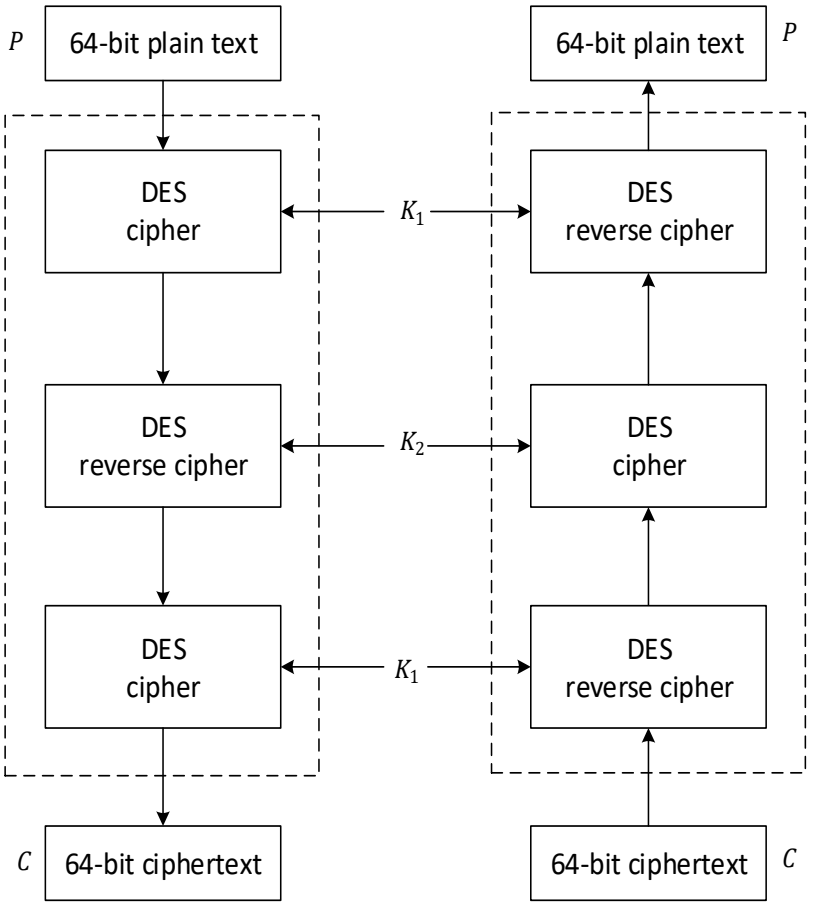
- 다중 DES
  - 이중 DES(2DES, Double DES)
    - 중간 일치 공격
      - 공격 과정
        1. 가능한 모든 K_1 후보로 평문 P를 암호화하여 중간값 M에 대한 테이블을 생성함
        2. 가능한 모든 K_2 후보로 암호문 C를 복호화하여 중간값 M과 비교함
        3. 두 중간 값이 일치하면 K_1, K_2 후보 쌍을 찾을 수 있음
      - 공격 복잡도
        - 2^{112}번의 전수조사가 아니라 2×2^{56}번의 검사로 공격 가능함

DES (Data Encryption Standard)

- 다중 DES
  - 삼중 DES(3DES, triple DES)
    - 정의
      - DES 알고리즘을 세 번 적용하여 보안성을 높인 방식임
        - 이중 DES가 중간 일치 공격에 취약한 한계를 보완하기 위해 사용됨
    - 알고리즘
      - 암호화 → 복호화 → 암호화 순서로 수행됨
    - 키 사용 방식
      - 두 키 방식: K_1, K_2, K_1 사용
      - 세 키 방식: K_1, K_2, K_3 사용

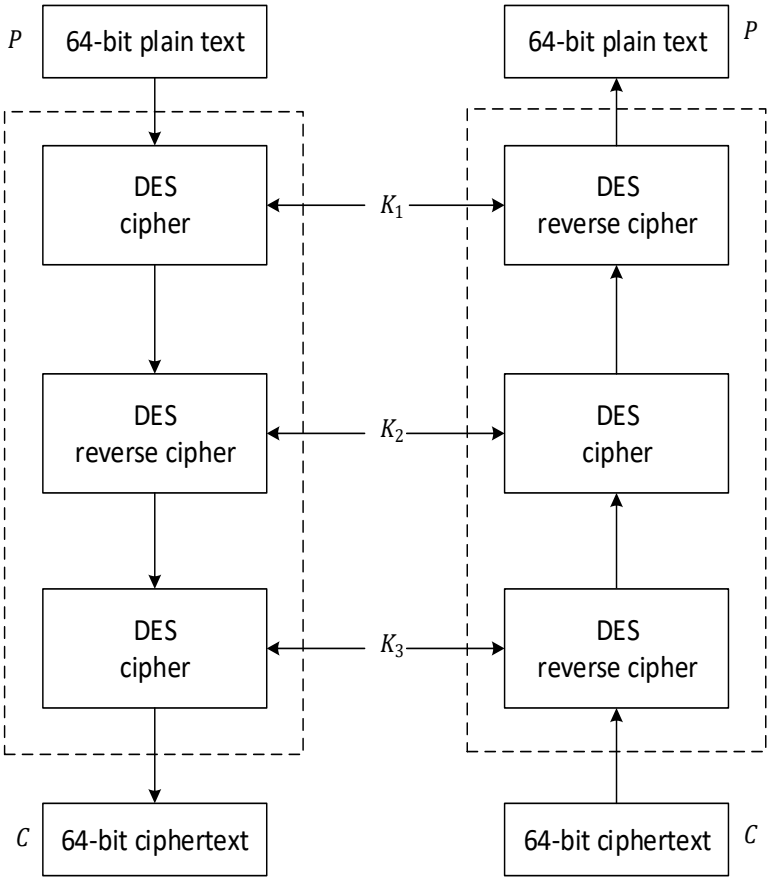
DES (Data Encryption Standard)

- 다중 DES
  - 삼중 DES(3DES, triple DES)
    - 두 개의 키를 갖는 삼중 DES
      - 정의
        - 두 개의 키 K_1, K_2를 사용하여 DES를 세 번 적용하는 방식임
      - 알고리즘
        - 암호화 → 복호화 → 암호화 순서의
        - 첫 번째와 세 번째 단계에서는 K_1을 사용함
        - 두 번째 단계에서는 K_2를 사용함
      - 관계식
        - $C = E_{K_1}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)$
      - 특징
        - 이중 DES의 중간 일치 공격 한계를 보완하기 위한 방식임
        - $K_1 = K_2$인 경우 단일 DES와 호환 가능함



DES (Data Encryption Standard)

- 다중 DES
  - 삼중 DES(3DES, triple DES)
    - 세 개의 키를 갖는 삼중 DES
      - 정의
        - 세 개의 키 K_1, K_2, K_3 를 사용하여 DES를 세 번 적용하는 방식임
      - 알고리즘
        - 암호화 → 복호화 → 암호화
        - K_1로 암호화하고, K_2 로 복호화한 뒤, K_3로 다시 암호화함
      - 관계식
        - $C = E_{K_3}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)$
      - 특징
        - 두 개의 키를 갖는 삼중 DES보다 키 공간이 더 큼
        - 보안성은 높아지지만 DES 연산을 세 번 수행하므로 처리 비용이 증가함



목 차

- 보충
- DES
  - 구조
  - 보안성 및 취약성
  - 다중DES
- AES
  - 데이터단위
  - 구조
  - 보안성 및 안전성

AES (Advanced Encryption Standard)

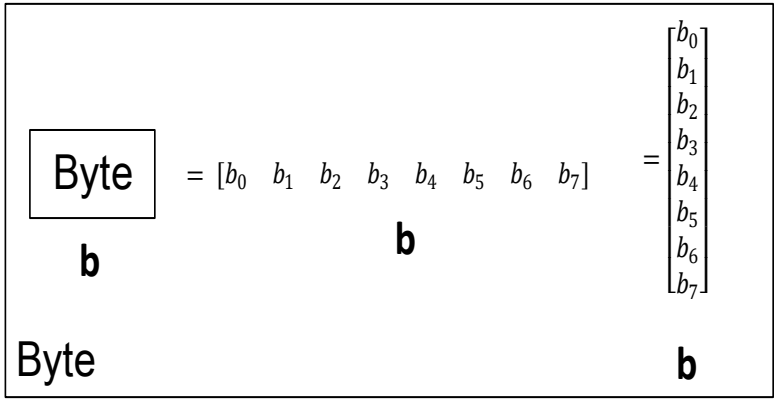
- 개요
  - 정의
    - 128비트 블록 단위의 데이터를 암호화하는 대칭 키 블록 암호 알고리즘임
  - 특징
    - 128비트 평문 블록을 128비트 암호문으로 변환함
    - 키 길이에 따라 AES-128, AES-192, AES-256으로 구분됨
    - 치환과 순열 연산을 기반으로 한 라운드 반복 구조를 사용함

AES (Advanced Encryption Standard)

- 개요
  - 역사
    - 1997년
      - 미국 NIST(National Institute of Standards and Technology)가 DES를 대체할 새로운 암호화 표준을 개발하기 위해 AES 공모를 발표
    - 2000년
      - NIST는 벨기에의 암호학자 Joan Daemen과 Vincent Rijmen이 개발한 Rijdael 알고리즘을 AES의 최종으로 선택
    - 2001년
      - Rijdael 알고리즘을 기반으로한 AES를 FIPS 197로공식 발표

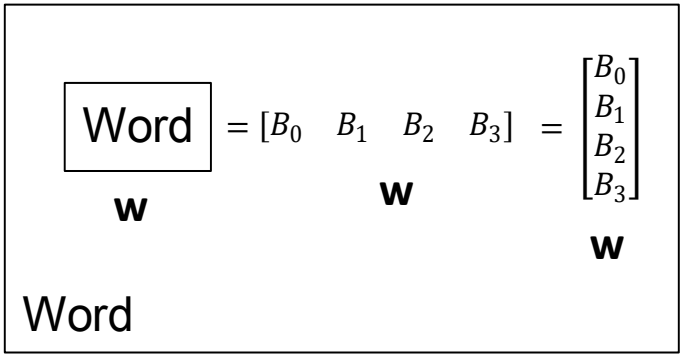
AES (Advanced Encryption Standard)

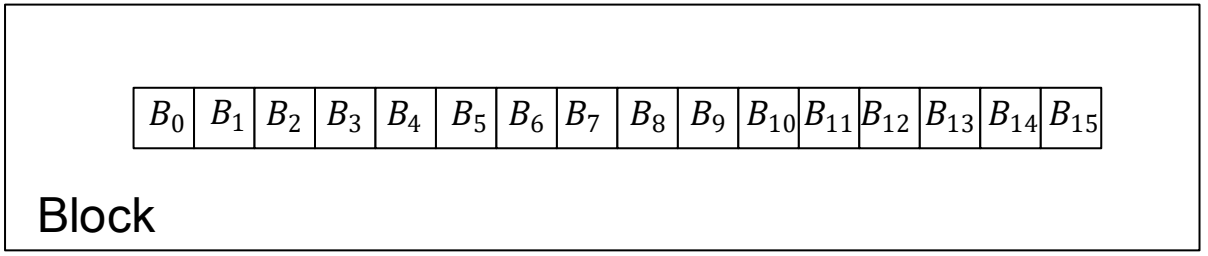
- 데이터 단위
  - 비트(Bit)
    - 0 또는 1을 나타내는 가장 작은 데이터 단위임
    - 표현 시, 소문자로 표현
      - e.g., b_1, b_2
  - 바이트(Byte)
    - 8개의 비트로 구성된 단위임
    - 비트를 원소로 하는 행 행렬(1×8) 또는 열 행렬(8×1)로 표현
    - 표현 시, 볼드 체 소문자로 표현
      - e.g., **b**



AES (Advanced Encryption Standard)

- 데이터 단위
  - 워드(Word)
    - 4개의 바이트로 구성된 32비트 단위임
    - 바이트를 원소로 하는 행 행렬 (1×4) 또는 열 행렬(4×1)로 표현
    - 표현 시, 볼드 체의 소문자 **w** 사용
      - e.g., **w**
  - 블록(Block)
    - 128비트(16바이트)로 이루어진 단위
    - 16바이트를 원소로 하는 행 행렬(1×16)로 표현





AES (Advanced Encryption Standard)

- 데이터 단위
  - 상태(State)
    - 16바이트를 원소로 가지는 4×4행렬로 표현
    - 각 열 4바이트를 하나의 워드로 볼 수 있음
      - State는 4개의 워드 w_0, w_1, w_2, w_3로 표현 가능함
    - 표현 시, S라 나타내며 임시 상태일 시에는 T로 표현

$$\mathbf{S} = \begin{bmatrix} \boldsymbol{S_{0,0}} & \boldsymbol{S_{0,1}} & \boldsymbol{S_{0,2}} & \boldsymbol{S_{0,3}} \\ \boldsymbol{S_{1,0}} & \boldsymbol{S_{1,1}} & \boldsymbol{S_{1,2}} & \boldsymbol{S_{1,3}} \\ \boldsymbol{S_{2,0}} & \boldsymbol{S_{2,1}} & \boldsymbol{S_{2,2}} & \boldsymbol{S_{2,3}} \\ \boldsymbol{S_{3,0}} & \boldsymbol{S_{3,1}} & \boldsymbol{S_{3,2}} & \boldsymbol{S_{3,3}} \end{bmatrix} = [w_0 \quad w_1 \quad w_2 \quad w_3]$$

State

AES (Advanced Encryption Standard)

- 데이터 단위
  - 예제 7.1

16개의 문자로 이루어진 블록이 어떻게 4×4행렬로 표현되는지 살펴본다.
주어진 문자가 “AES uses a matrix” 라고 가정하자

- 마지막에 가짜 문자를 추가하여 16개의 문자를 얻음

A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z

- 각 문자를 00과 25 사이의 정수로 나타내고 이를 16진수로 표현함

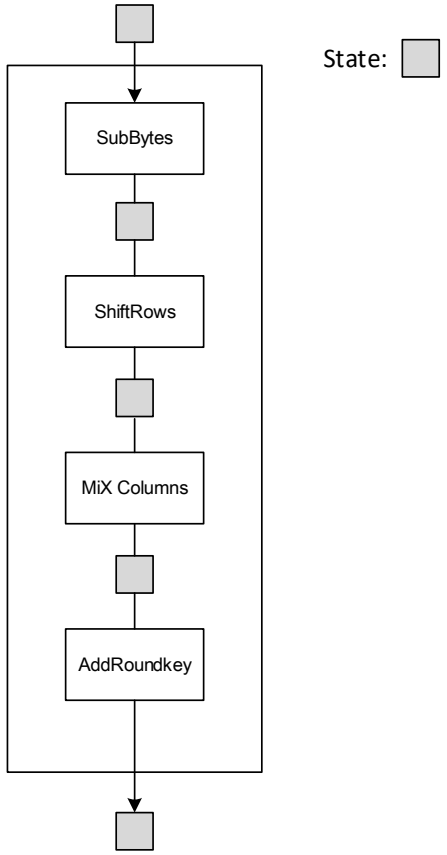
00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19

- 왼쪽 문자열부터 채움

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$

AES (Advanced Encryption Standard)

- 라운드(Round)
  - AES의 라운드 함수는 4가지 형태의 변환을 사용함
    - 변환
      - 대치(Substitution)
        - 입력 값을 정해진 규칙에 따라 다른 값으로 바꾸는 변환
        - SubBytes, InvSubBytes
      - 치환(Permutation)
        - 데이터 값 자체는 바꾸지 않고, 위치만 재배열하는 변환
        - ShiftRows
      - 뒤섞음(Mixing)
        - 여러 바이트 값을 서로 섞어서, 한 바이트의 변화가 여러 바이트에 영향을 주도록 만드는 변환
        - Mix Columns, InvMix Columns
      - 키 덧셈(Key-Adding)
        - 상태(State) 배열과 라운드 키(Round Key)를 XOR 연산하여 결합하는 변환
        - AddRoundKey

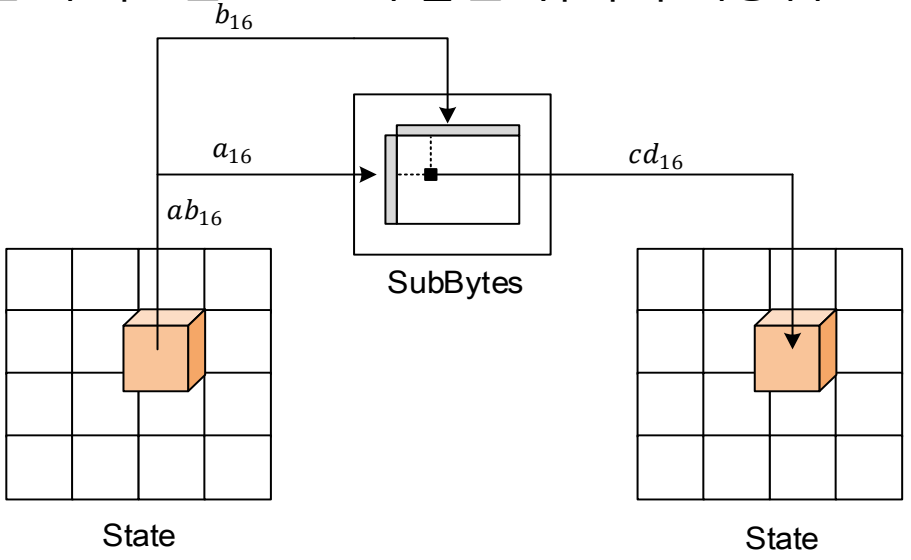


AES (Advanced Encryption Standard)

- 라운드(Round)
  - 대치(Substitution)
    - 부분바이트(SubBytes)
      - 정의
        - 상태 배열(State)의 각 바이트를 S-Box를 이용해 다른 바이트 값으로 대치하는 변환
      - 특징
        - 1바이트씩 독립적으로 변환함
        - 입력과 출력 사이의 단순한 관계를 없애 분석을 어렵게 함
        - 평문과 암호문 사이의 관계를 복잡하게 만듦
        - 복호화 과정에서는 InvSubBytes를 사용함

AES (Advanced Encryption Standard)

- 라운드(Round)
  - 대치(Substitution)
    - 부분바이트 (SubBytes)
      - 부분바이트 변환 표 참조 과정
        1. 입력 바이트를 16진수 두 자리로 표현함
           - ab_{16}
        2. 앞자리 a는 S-Box의 행을 선택하고 뒷 자리 b는 S-Box의 열을 선택함
        3. 선택된 행과 열이 만나는 값 cd_{16} 이 새로운 출력 바이트가 됨
        4. 변환된 바이트는 State의 같은 위치에 저장됨



AES (Advanced Encryption Standard)

- 라운드(Round)
  - 대치(Substitution)
    - 부분바이트 (SubBytes)
      - $GF(2^8)$을 이용한 변환 방법
        - 변환 원리
          - 입력 바이트를 $GF(2^8)$의 원소로 보고 역원을 구함
          - 구한 역원에 *아핀 변환을 적용함
          - 최종 8비트 값이 SubBytes 출력값이 됨
        - 구성 요소
          - S_r, c: State의 r행 c열에 위치한 입력 바이트임
          - $(S_r, c)^{-1}$: 입력 바이트의 곱셈 역원임
          - X: AES에서 정해진 고정 행렬임
          - y: AES에서 정해진 고정 벡터임
          - d: SubBytes 변환 후 생성되는 출력 바이트임
        - 관계식
          - SubByte: $d = X(S_r, c)^{-1} \oplus y$
          - InSubByte: $[X^{-1}(d \oplus y)]^{-1} = [X^{-1}(X(S_r, c)^{-1} \oplus y \oplus y)]^{-1} = S_r, c$

*아핀 변환:역원으로 바뀐 값을 AES가 정한 방식으로 한 번 더 섞어 최종 S-Box 출력값을 만드는 과정임

AES (Advanced Encryption Standard)

- 라운드(Round)
  - 대치(Substitution)
    - 부분바이트 (SubBytes)
      - 예제 7.3

16진수 $0C$가 SubByte 통해 FE로 계산되는 과정을 계산해라

- SubBytes
  1. $GF(2^8)$에서 $0C_{16}$의 곱셈 역원을 구함
     $(0C_{16})^{-1} = B0_{16} = 10110000_2$
  2. 관계식 $d = X(S_r, c)^{-1} \oplus y$ 를 적용함
     $d = X(B0_{16}) \oplus y$
  3. 변환 결과는 FE_{16}임
- 결과
  - $11111110_2 = FE_{16}$
  - 따라서 SubBytes$(0C) = FE$임

AES (Advanced Encryption Standard)

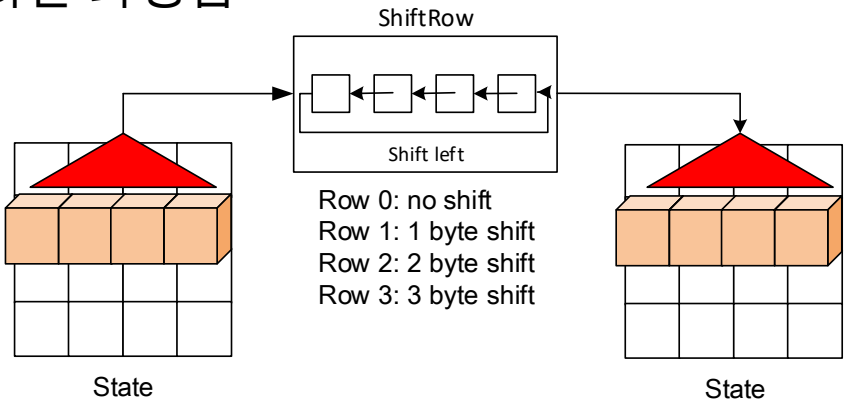
- 라운드(Round)
  - 대치(Substitution)
    - 부분바이트 (SubBytes)
      - 예제 7.3

InSubByte를 통해 다시 $0C$로 계산하는 과정을 설명하라(역부분바이트 관계식)

- InvSubBytes
  1. 입력 바이트를 $d = FE_{16}$으로 둠
  2. 관계식 $[X^{-1}(d \oplus y)]^{-1}$을 적용함
  3. 먼저 FE_{16}에 역아핀 변환을 적용함
     - 결과는 $B0_{16}$
  4. $GF(2^8)$에서 $B0_{16}$의 곱셈 역원을 구함
     - 결과는 $0C_{16}$
- 결과
  - InvSubBytes(FE_{16}) = $0C_{16}$
  - SubBytes와 InvSubBytes는 서로 역변환 관계임

AES (Advanced Encryption Standard)

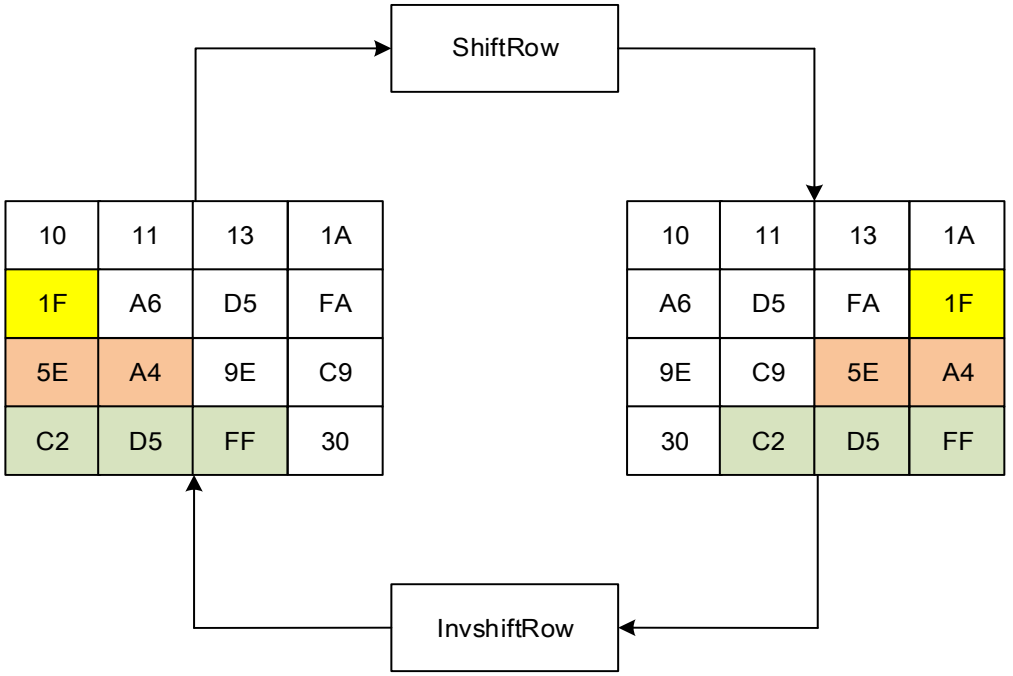
- 라운드(Round)
  - 치환(Permutation) 또는 이동 연산(Shifting)
    - ShiftRows
      - 정의
        - State의 각 행을 왼쪽으로 순환 이동하는 과정임
      - 이동 방식
        - 0번째 행: 이동하지 않음
        - 1번째 행: 왼쪽으로 1바이트 이동함
        - 2번째 행: 왼쪽으로 2바이트 이동함
        - 3번째 행: 왼쪽으로 3바이트 이동함
      - 특징
        - 바이트 값은 변하지 않고 행렬 상의 위치만 이동함
        - 이동된 바이트는 사라지지 않고 반대쪽 끝으로 이동함
        - 바이트의 위치를 재배열하여 한 열의 값들이 여러 열로 퍼지게 하기 때문에 확산성을 확보함



AES (Advanced Encryption Standard)

- 라운드(Round)
  - 치환(Permutation) 또는 이동 연산(Shifting)
    - 예제 7.4

다음 상태(State)에 대해 ShiftRows 변환을 수행하시오.

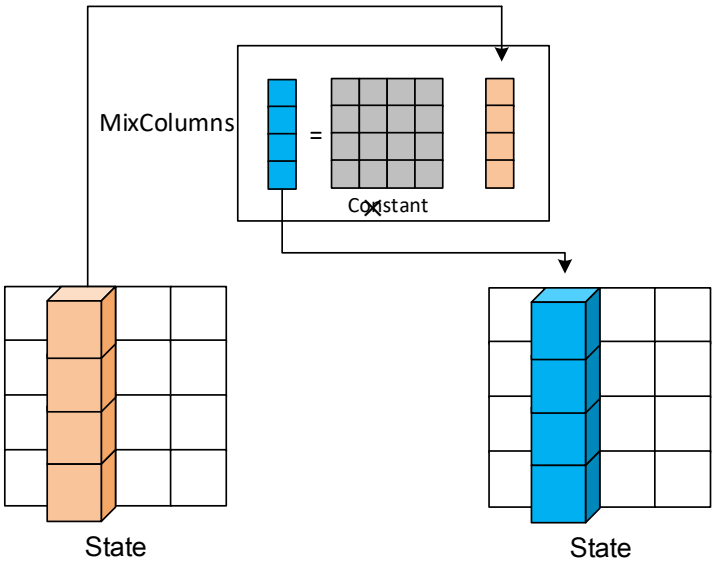


AES (Advanced Encryption Standard)

- 라운드(Round)
  - 뒤섞음(Mixing)
    - MixColumns
      - 정의
        - State의 각 열을 고정 행렬과 곱하여 새로운 열 값을 생성함
      - 변환 방식
        - 각 열을 AES에서 정한 고정 행렬과 곱함
        - 이때 곱셈은 $GF(2^8)$에서 수행함
        - 계산 결과가 8비트를 넘으면 기약다항식으로 나머지 연산을 하여 8비트 값으로 정리함
      - 특징
        - 같은 열 안의 4바이트가 서로 영향을 주도록 만듦
        - AES의 확산 효과를 높이는 역할임
        - 키나 평문에 따라 고정 행렬은 바뀌지 않음
        - InvMixColumns는 MixColumns의 역변환임
          - 암호화 과정에서 섞인 State의 각 열을 복호화 과정에서 다시 되돌리는 변환임

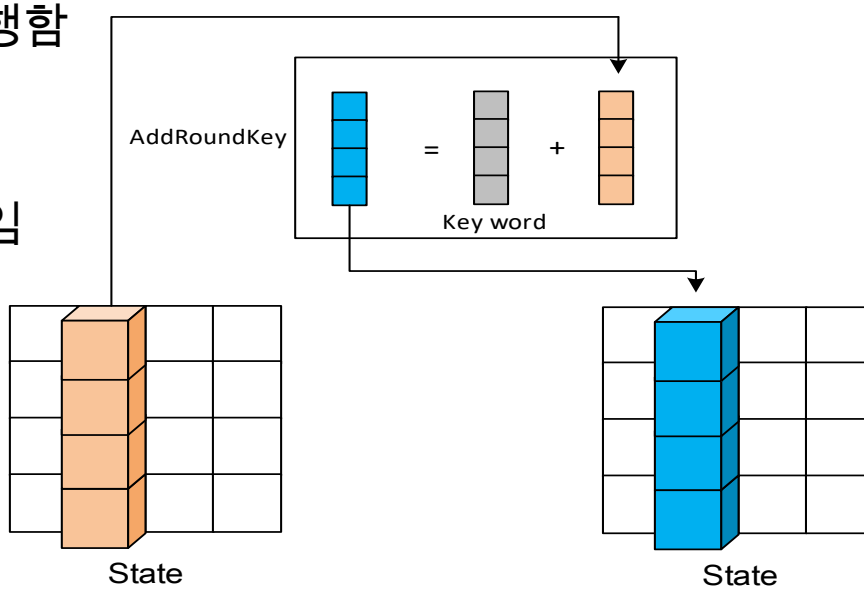
$$\begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix}$$

AES 고정 행렬



AES (Advanced Encryption Standard)

- 라운드(Round)
  - 키 덧셈(Key-Adding)
    - AddRoundKey
      - 정의
        - State의 각 열에 라운드 키 워드를 XOR하는 과정임
      - 변환 과정
        - 키 확장 과정을 통해 라운드 키를 생성함
        - State의 각 열과 라운드 키 워드를 같은 위치끼리 대응시킴
        - 대응되는 바이트끼리 XOR 연산을 수행함
        - XOR 결과가 새로운 State 값이 됨
      - 특징
        - XOR 연산이므로 자기 자신의 역변환임
        - 키와 암호문 사이의 관계를 복잡하게 만들어 혼돈 효과에 기여함
        - XOR은 자기 자신의 역연산이므로 복호화에서도 같은 연산을 사용함

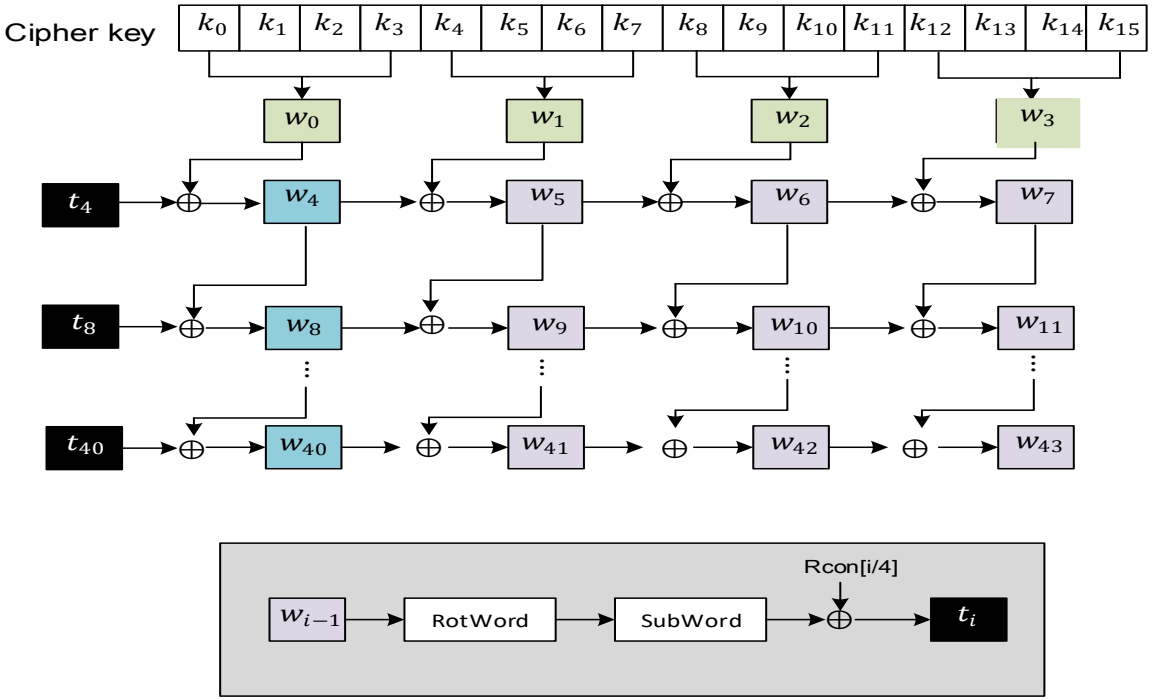


AES (Advanced Encryption Standard)

- 암호화 과정
  - (1단계)
    - 평문 블록을 State 배열로 초기화
  - (2단계)
    - 초기 암호 키를 여러 라운드 키로 확장
  - (3단계)
    - 라운드 수행
      - SubBytes
      - ShiftRows
      - MixColumns
      - AddRoundKey

AES (Advanced Encryption Standard)

- 암호화 과정
  - 키 확장
    - 하나의 비밀 키로부터 여러 개의 라운드 키를 생성함
    - 생성된 라운드 키는 각 라운드의 AddRoundKey에서 사용됨
    - AES-128은 총 44개의 워드를 생성함

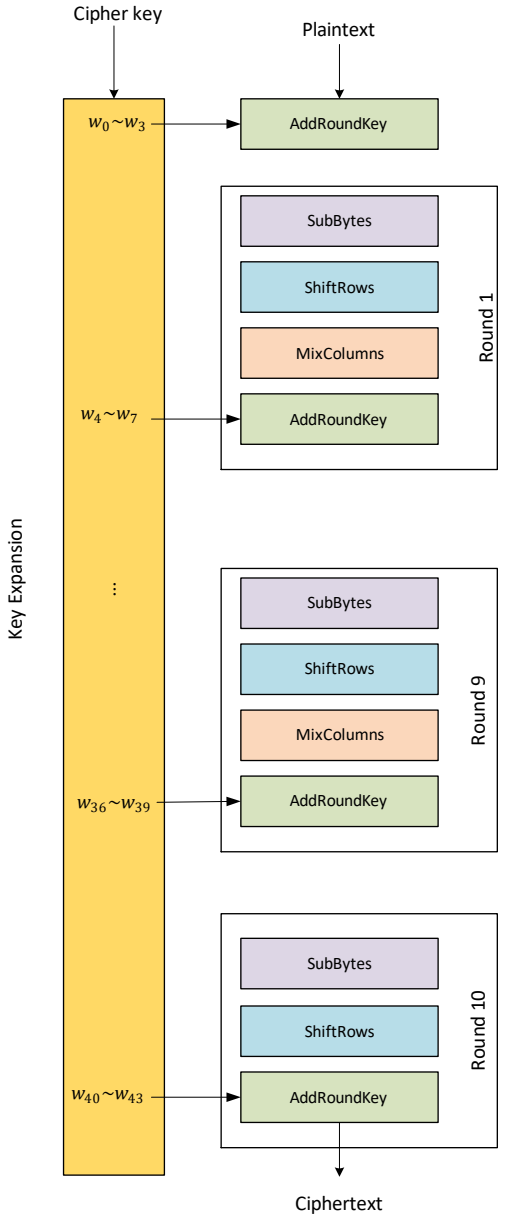


AES (Advanced Encryption Standard)

- 암호화 과정
  - 키 확장
    - RotWord
      - 워드 안의 바이트 위치를 왼쪽으로 한 칸 순환 이동하는 연산임
    - SubWord
      - RotWord 결과의 각 바이트를 S-Box를 이용해 다른 값으로 바꾸는 연산임
    - Rcon
      - 라운드마다 다르게 사용되는 고정 상수값임
    - ㄴ 과정
      - $i \, mod \, 4 \neq 0$인 경우
        - $\boldsymbol{w_i = w_{i-1} \oplus w_{i-4}}$
      - $i \, mod \, 4 = 0$인 경우
        - RotWord, SubWord, RCon을 적용함
          - $t = SubWord(RotWord(w_{i-1})) \oplus \mathrm{RCon}_{i/4}$

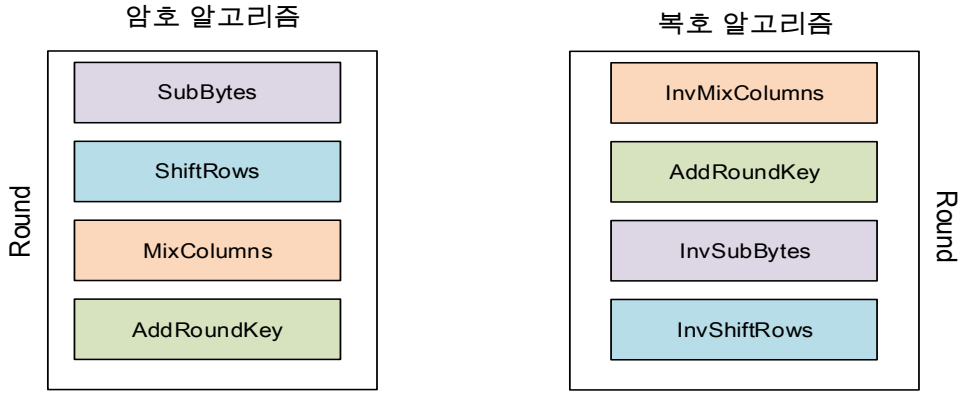
AES (Advanced Encryption Standard)

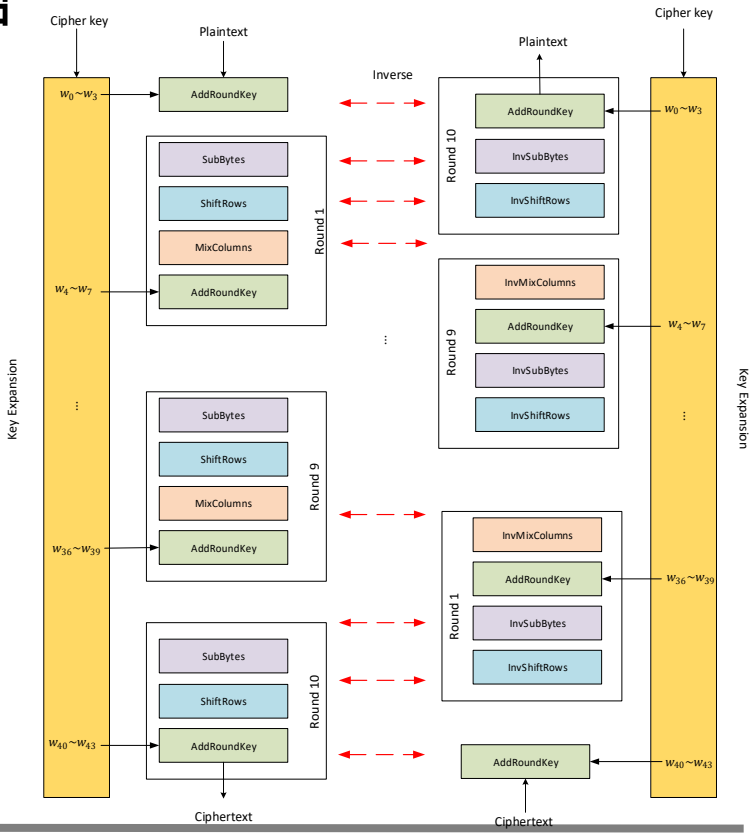
- 암호화 과정
  - (3단계)라운드 수행
    - 초기 라운드
      - AddRoundKey
    - 일반 라운드
      - SubBytes
      - ShiftRows
      - MixColumns
      - AddRoundKey
    - 최종 라운드
      - SubBytes
      - ShiftRows
      - AddRoundKey



AES (Advanced Encryption Standard)

- 복호화 과정
  - 복호 알고리즘은 암호 알고리즘의 역함수임
  - SubBytes와 ShiftRows
    - SubBytes는 값 변경, ShiftRows는 위치 이동만 수행함 두 연산은 서로 독립적이므로 순서 변경이 가능함
  - MixColumns와 AddRoundKey
    - MixColumns는 열의 바이트 값을 서로 섞는 연산임
    - AddRoundKey의 순서를 바꾸려면 라운드 키도 함께 변형해야 함



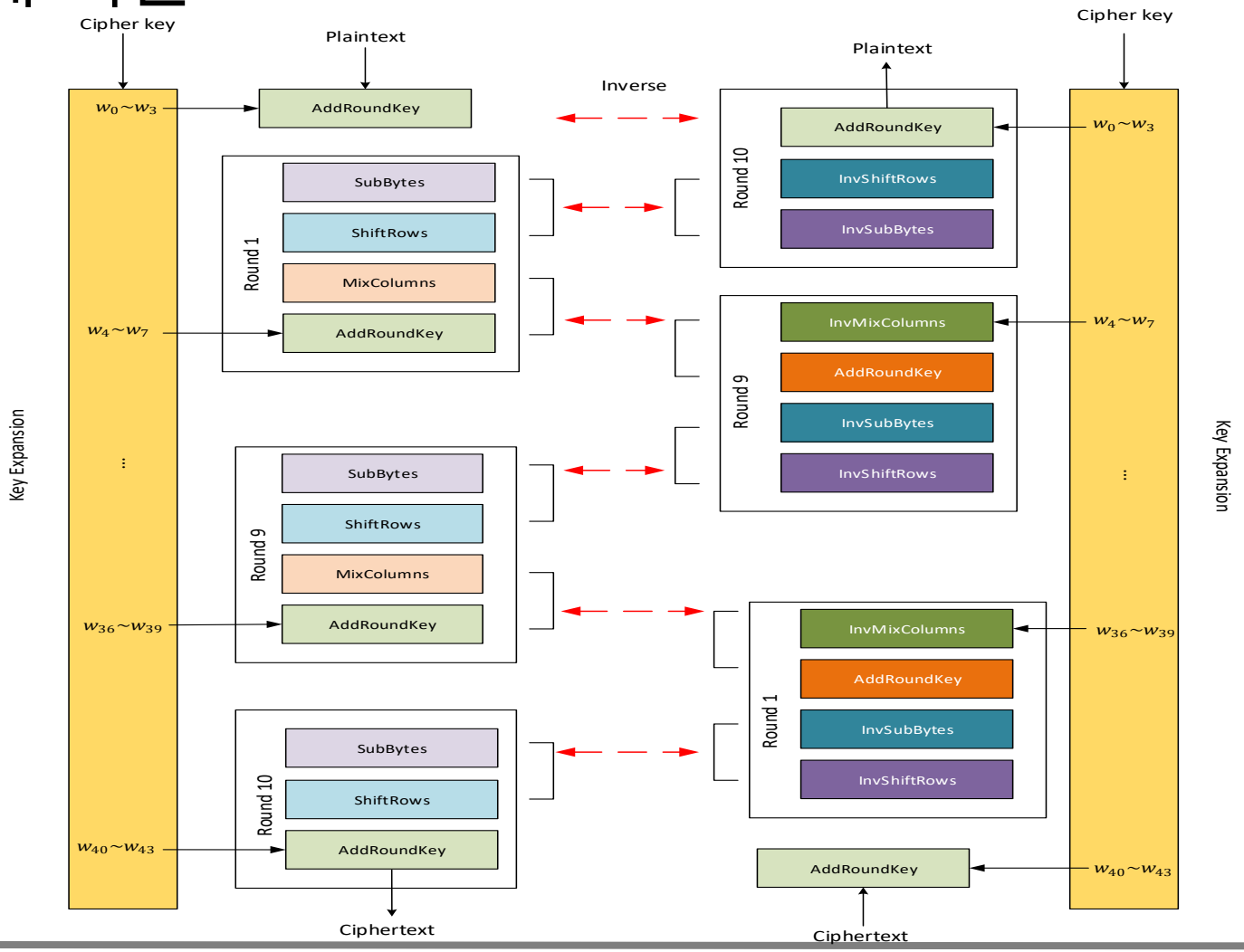


AES (Advanced Encryption Standard)

- 다른 설계 버전
  - 암호화와 복호화의 연산 흐름을 비슷하게 만들기 위해 사용함
    - 같은 구조의 회로 또는 프로그램 코드를 재사용하기 쉬움
  - 기존 방식은 암호화 라운드를 역순으로 복호화했지만, 다른 설계 버전은 암호화와 유사한 라운드 구조를 사용하도록 변경한 방식임

AES (Advanced Encryption Standard)

- 구조
  - 다른 설계 버전



AES (Advanced Encryption Standard)

- 분석
  - 안정성
    - 전수조사 공격
      - AES는 128, 192, 256비트 키를 사용함
      - DES보다 훨씬 큰 키 공간을 가지므로 전수조사 공격에 강함
    - 통계적 공격
      - SubBytes, ShiftRows, MixColumns의 조합으로 혼돈과 확산을 제공함
      - 암호문의 통계적 패턴 분석을 어렵게 만듦
    - 차분 공격과 선형 공격
      - DES 이후 알려진 공격을 고려하여 설계됨
      - 현재까지 실용적인 공격은 알려져 있지 않음

AES (Advanced Encryption Standard)

- 보안성
  - 키 길이
    - AES는 128, 192, 256비트 키를 지원함
    - 키 길이가 길수록 가능한 키 후보 수가 증가함
    - 전수조사 공격에 필요한 시간이 크게 증가함
  - 라운드 수
    - AES-128은 10라운드
    - AES-192는 12라운드
    - AES-256은 14라운드
    - 키 길이가 길어질수록 라운드 수가 증가하므로 혼돈과 확산이 보장됨

AES (Advanced Encryption Standard)

- 보안성
  - 보안성과 처리 속도
    - 라운드 수가 많아지면 분석 공격에 대한 저항성은 높아짐
    - 라운드 수가 증가하면 연산량도 함께 증가함
    - AES는 보안성과 처리 효율을 함께 고려하여 라운드 수를 정함

Thanks!

김 민 식(ms6127@naver.com)