2024/07/11, 2024 보안 기초 세미나

암호학과 네트워크 보안

-6장 DES, 7장 AES-

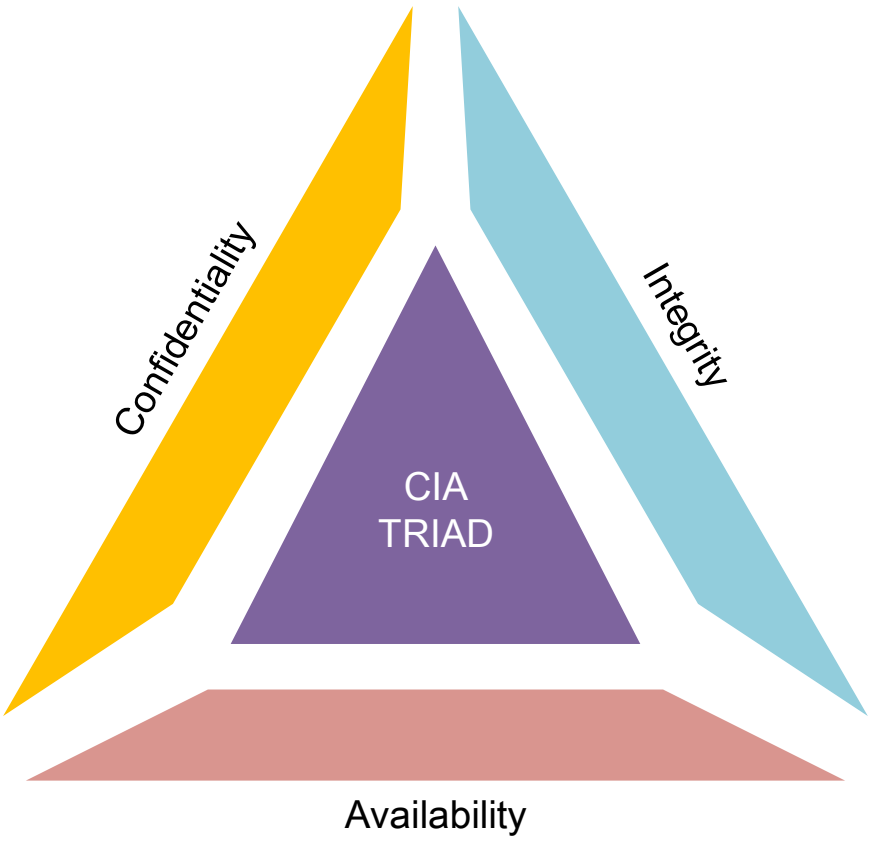
김 혜 정(hyejeong@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

보충

- 보안 목표
  - CIA Triad
    - 기밀성(Confidentiality)
      - 인가된 사용자만 정보 자산에 접근할 수 있는 원칙
    - 무결성(Integrity)
      - 인가된 사용자 이외에 정보를 변경되지 않도록 하는 원칙
    - 가용성(Availability)
      - 사용자가 필요 시 언제든 정보에 접근할 수 있도록 하는 원칙



보안 개요

- 보안 목표가 침해된 경우
  - 기밀성이 침해된 경우
    - 암호화되지 않은 데이터를 전달할 때, 제 3자가 해당 데이터 패킷을 가로채 도청 또는 분석할 경우
  - 무결성이 침해된 경우
    - 공격자가 이메일 서버에 침입하여 발신자의 이메일 주소를 위조하고, 보내고자하는 이메일의 내용을 조작하여 수신자에게 전달한 경우
  - 가용성이 침해된 경우
    - 특정 웹 사이트가 DDoS 공격을 받아 서버가 다운되어 고객들이 사이트에 접속하지 못하는 경우

보안 개요

- 암호
  - 비밀을 유지하기 위하여 당사자끼리만 알 수 있도록 꾸민 약속 기호 또는 알고리즘
  - 종류(1/3)
    - 대칭-키 암호화 (Symmetric-key Encipherment)
      - 동일한 키를 사용하여 데이터를 암복호화하는 방식
      - 두 가지 방식으로 나뉨
        - Stream 방식: 평문을 비트 단위로 분할하여 암호화
        - Block 방식: 평문을 블록단위로 분할하여 암호화
      - 키를 교환해야한다는 문제와 사람이 증가할수록 키 관리가 어려워짐
      - e.g., DES(Data Enctyption Standard), RC4(Ron's Code 4)

보안 개요

- 암호
  - 종류(2/3)
    - 비대칭-키 암호화 (Asymmetric-key Encipherment)
      - 서로 다른 키를 사용하여 암복호화하는 방식
      - 두 개의 키, 개인키와 공개키가 쌍을 이룬 형태
        - 개인 키(Private Key): 외부에 절대 노출되어서는 안되는 키
        - 공개 키(Public Key): 공개적으로 개방되어있는 키
      - 비밀 키 배송 문제를 해결
      - e.g., RSA(Rivest Shamir Adleman), DSA(Digital Signature Algorothm)
    - 해싱(Hashing)
      - 다양한 길이의 메시지를 해시함수(Hash Function)로 고정된 길이의 메시지로 압축
      - 해싱 완료시, 해시값을 통해 본래 문자열을 알 수 없으며 변조 여부만 확인 가능
      - 동일 문자열은 동일 해시 알고리즘을 사용 시 반드시 동일한 해시 값을 출력

보안 개요

- 암호
  - 대칭키와 비대칭키의 효과
    - 대칭 키
      - 알고리즘 구현이 간단하고 처리 속도가 빨라 데이터 전송, 파일이나 데이터베이스 암호화에 주로 쓰임
      - 키 길이가 짧고, 연산의 복잡성이 낮아 쉽게 해독될 수 있음
    - 비대칭 키
      - 연산의 복잡성이 높아 안전성이 강해 공인인증기관에서 서명이나 인증서 발급 시 사용
      - 키 길이가 길고, 복잡한 수학적 연산을 이용하여 구현되므로 처리 시간이 김

대수 구조

- 군(G, Group)
  - 부분 군(H, Subgroup)
    - 예제) 군 $H = < Z_{10}, + >$ 가 군 $G = < Z_{12}, + >$ 의 부분 군인가?
      - 라그랑지 정리를 위배함
        - 라그랑지 정리에 의해 군 G의 부분 군이 가지는 위수는 각 1, 2, 3, 4, 6, 12개만 가능하다는 사실을 알 수 있음
      - 군 H는 $mod\ 10$의 연산을, 군 G는 $mod\ 12$의 연산을 말하기 때문

목 차

DES(Data Encryption Standard)

- 개요
  - 정의
    - 64비트의 평문을 64비트의 암호문으로 암호화하는 대칭 키 블록 암호
    - 키 길이는 56비트 암호 키를 암호화와 복호화에 동일하게 사용

DES

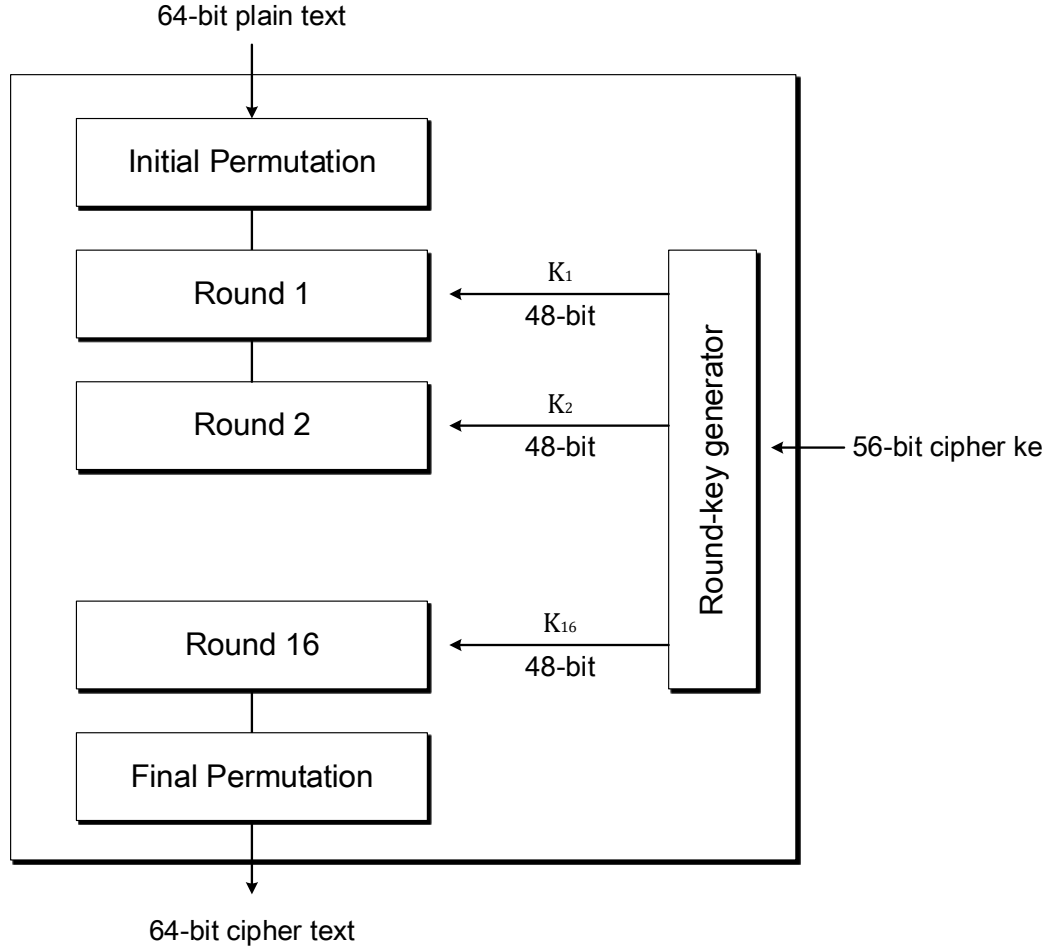
- 개요
  - 역사
    - 1973년
      - 미국 NBS(National Bureau of Standards)가 국가적으로 사용할 대칭 키 암호시스템 제안 요청서 발표
    - 1975년
      - IBM이 개발한 DES가 채택
    - 1977년
      - 연방정부의 FIPS 46(정보 처리용 표준 암호 기술)이란 이름으로 발표

DES

- 개요
  - 비판
    - 56비트의 작은 키 길이는 전수조사 공격에 취약
      - 키 공간이 $2^{56}(7.2 \times 10^{16})$, 72조개의 키를 의미
      - DES가 설계된 70년대에는 키 공간이 충분히 컸으나, 컴퓨터 성능이 향상되면서 전수조사 공격이 실현 가능해질 것을 우려
    - DES의 구성 요소 중 하나인 S-박스에 약점이 있어 키 없이도 메시지를 복호화할 수 있을 것이라는 의심
      - DES는 IBM에 의해 개발되었으나, 미국 국가안보국 (NSA, National Security Agency)의 조언을 받아 최종적으로 수정되어 일부 암호학자들이 NSA가 DES에 의도적으로 약한 S-박스를 포함시켰을 가능성을 의심
      - DES의 S-박스 설계 기준이 공개되지 않아 내부구조를 신뢰할 수 있는지에 대한 의문 제기

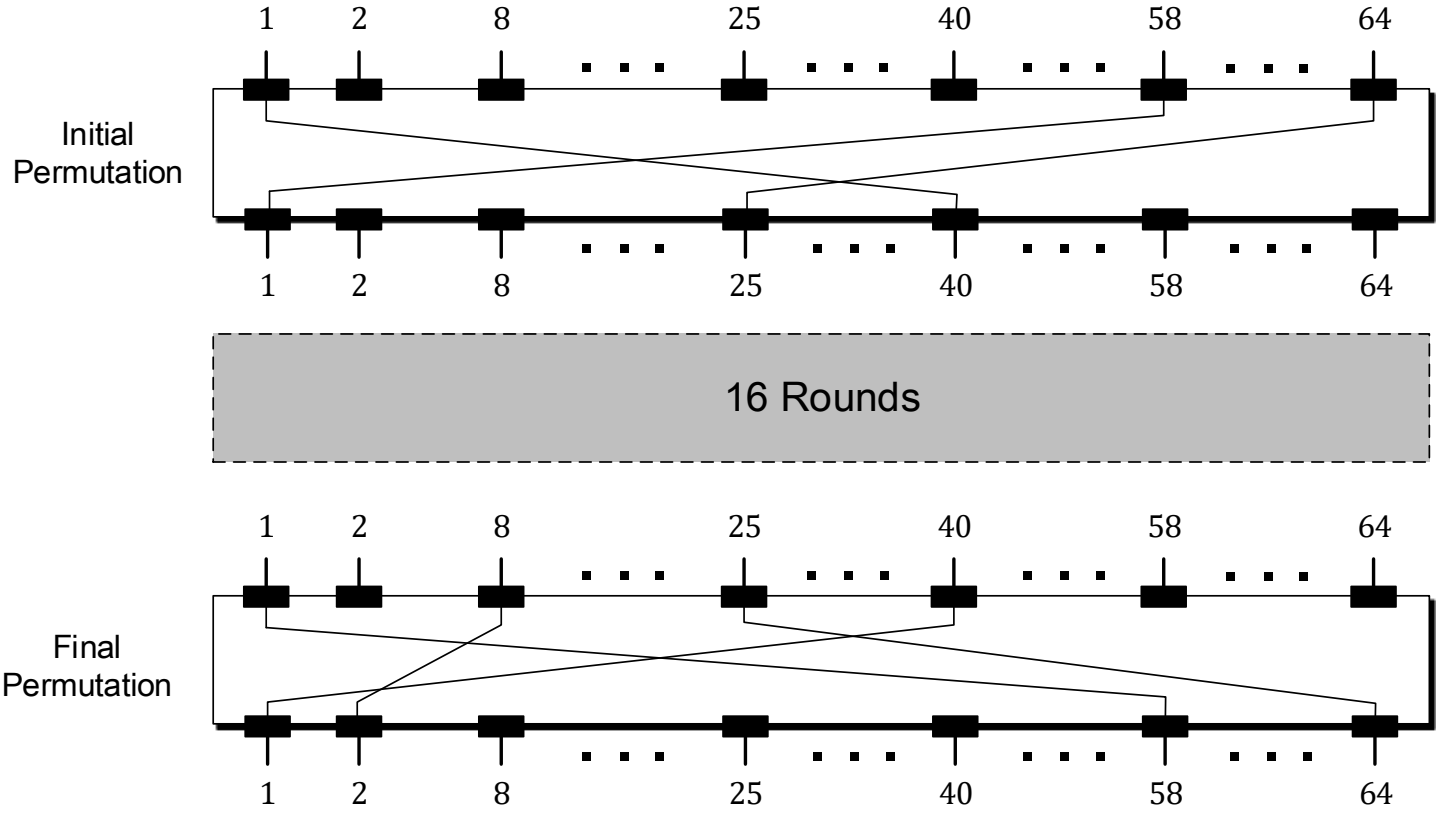
DES

- 구조
  - 초기 전치(IP, Initial Permutation)과 최종 전치(FP, Final Permutation)이라고 부르는 두 개의 P-박스와 16개의 라운드로 구성



DES

- 구조
  - 초기 전치과 최종 전치 (1/2)
    - 각 전치는 64비트를 입력 받아 미리 정의된 규칙(단순 P-박스)에 의해 재배열



DES 구조

- 구조
  - 초기 전치과 최종 전치 (2/2)
    - 초기 전치(IP)과 최종 전치(FP)은 서로 역 관계
      - 초기 전치의 i번째 비트가 j번째 비트로 이동하면, 최종 전치에서 j번째 비트가 i번째 비트로 이동
      - e.g., IP[58] = 1, FP[1] = 58 (i=58, j= 1)
    - 키가 없는 단순 순열

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

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

DES 구조

- 초기 전치과 최종 전치
  - 예제 6.1

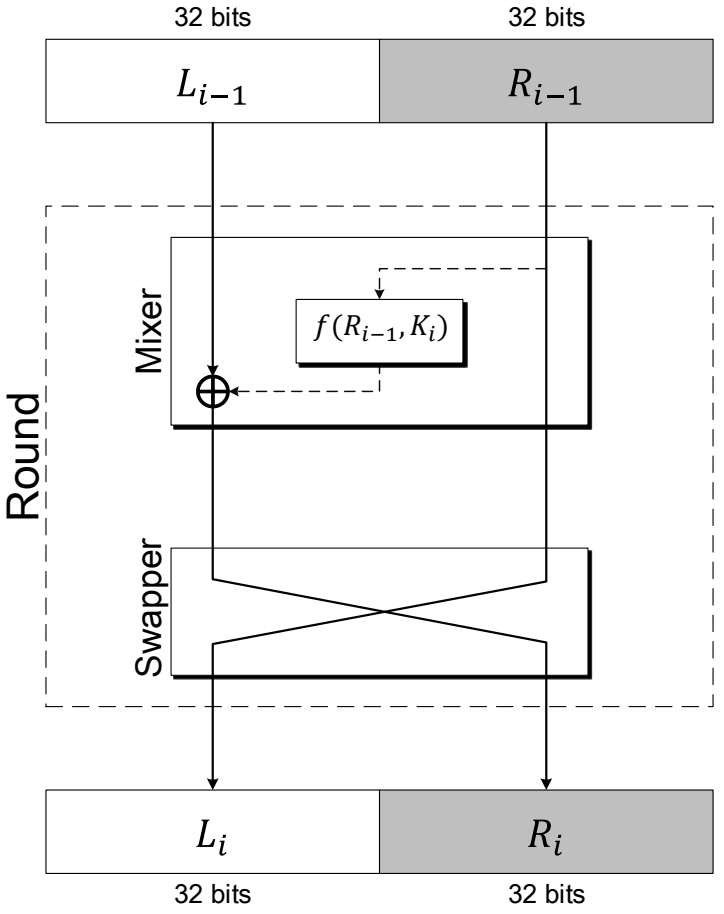
입력 값이 다음과 같은 16진수일 때, 초기 전치에 의한 출력 값을 구하시오.
0x0002 0000 0000 0001

- 16진수 입력 값을 2진수로 바꾸면,
  0000000000000010 0000000000000000
  0000000000000000 0000000000000001
- 15번째와 64번째 비트만 1임
- 표에 의해 전치 시키면 IP[64] = 25, IP[15] = 63
  0000000000000000 0000000010000000
  0000000000000000 0000000000000010
- 전치 값을 16진수로 바꾸면,
  0x0000 0080 0000 0002

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

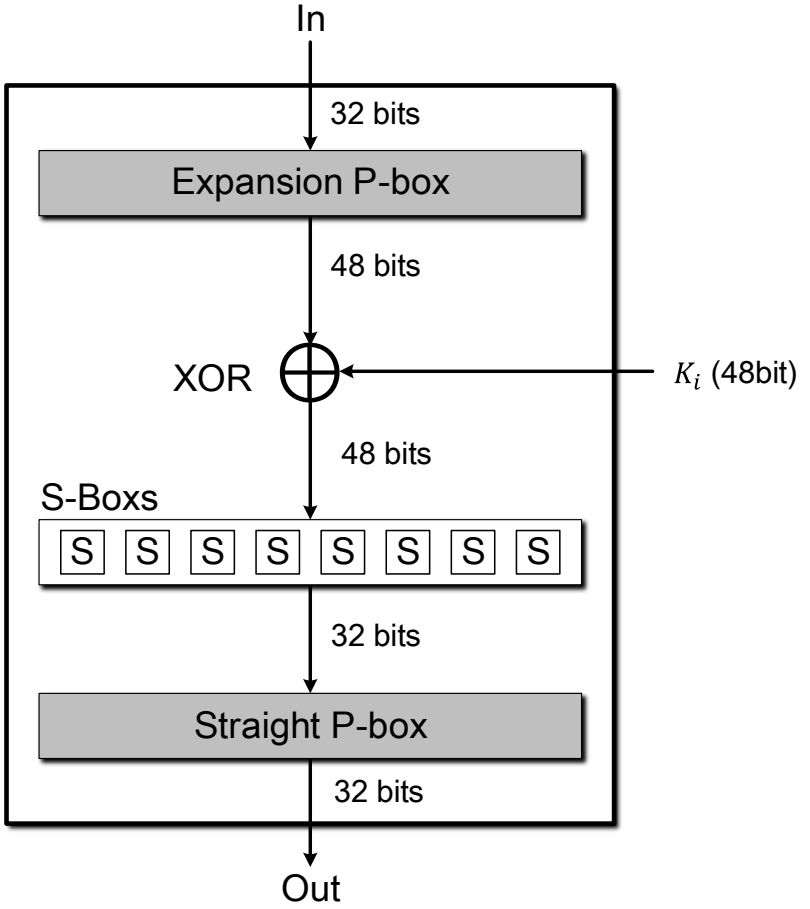
DES

- 구조
  - 라운드(Round)
    - 이전 라운드(또는 초기 전치)의 출력 값 L_{i-1}과 R_{i-1}을 입력받아, 다음 라운드(또는 최종 전치)에 입력으로 적용될 L_i과 R_i을 생성하는 과정
    - 구성
      - 혼합기(Mixer)
      - DES 함수($f(R_{i-1}, K_i)$)
      - 스와퍼(Swapper)



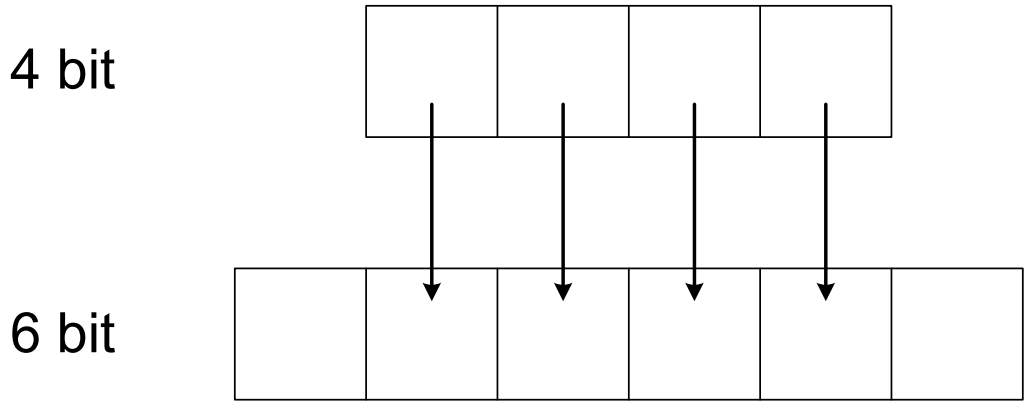
DES

- 구조
  - DES 함수 (1/7)
    - 확장 P-박스, 키 XOR, 8개의 S-박스, 단순 P-박스로 구성
    - 단계
      - (1단계)
        - 확장 P-박스를 거쳐 32비트 블록을 48비트로 확장
      - (2단계)
        - 라운드키와 48비트 블록을 XOR 연산
      - (3단계)
        - S-박스를 거쳐 48비트 블록을 32비트로 축약
      - (4단계)
        - 단순 P-박스를 걸쳐 전치



DES

- 구조
  - DES 함수 (2/7)
    - (1단계) R_{i-1}의 32비트가 확장 P-박스를 거쳐 48비트로 확장
      - 확장 P-박스(E table, Expansion/Permutation)
        - R_{i-1}의 32비트를 48비트로 늘려 전치 시키는 역할
        - R_{i-1}은 8개의 4비트로 나누어져 확장 전치를 거친 뒤 6비트 값으로 확장됨
        - 4비트 입력 값에 대해, 비트 1, 2, 3, 4는 각각 출력 비트 2, 3, 4, 5가 됨



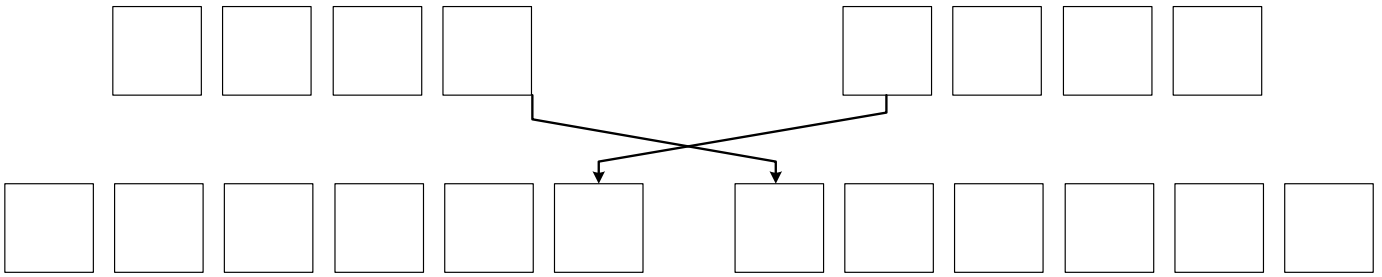
E table					
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

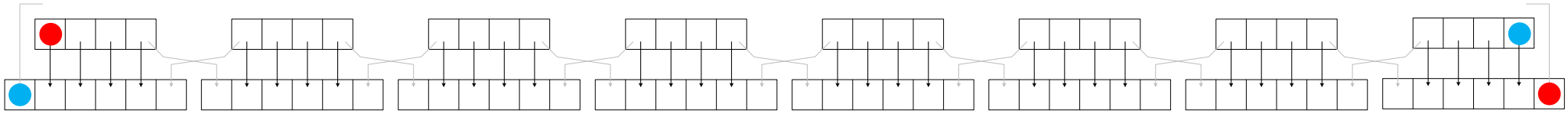
DES

- 구조
  - DES 함수 (3/7)
    - (1단계) R_{i-1}의 32비트가 확장 P-박스를 거쳐 48비트로 확장
      - 확장 P-박스(E table, Expansion/Permutation)
        - 출력 값 1번째 비트는 이전 4비트 입력 값의 4번째 비트로부터, 출력 값 여섯 번째 비트는 다음 4비트 입력 값의 첫 번째로부터 나옴
        - 1번째 4비트와 8번째 4비트는 인접해 있다고 간주해서 도출

E table ▸

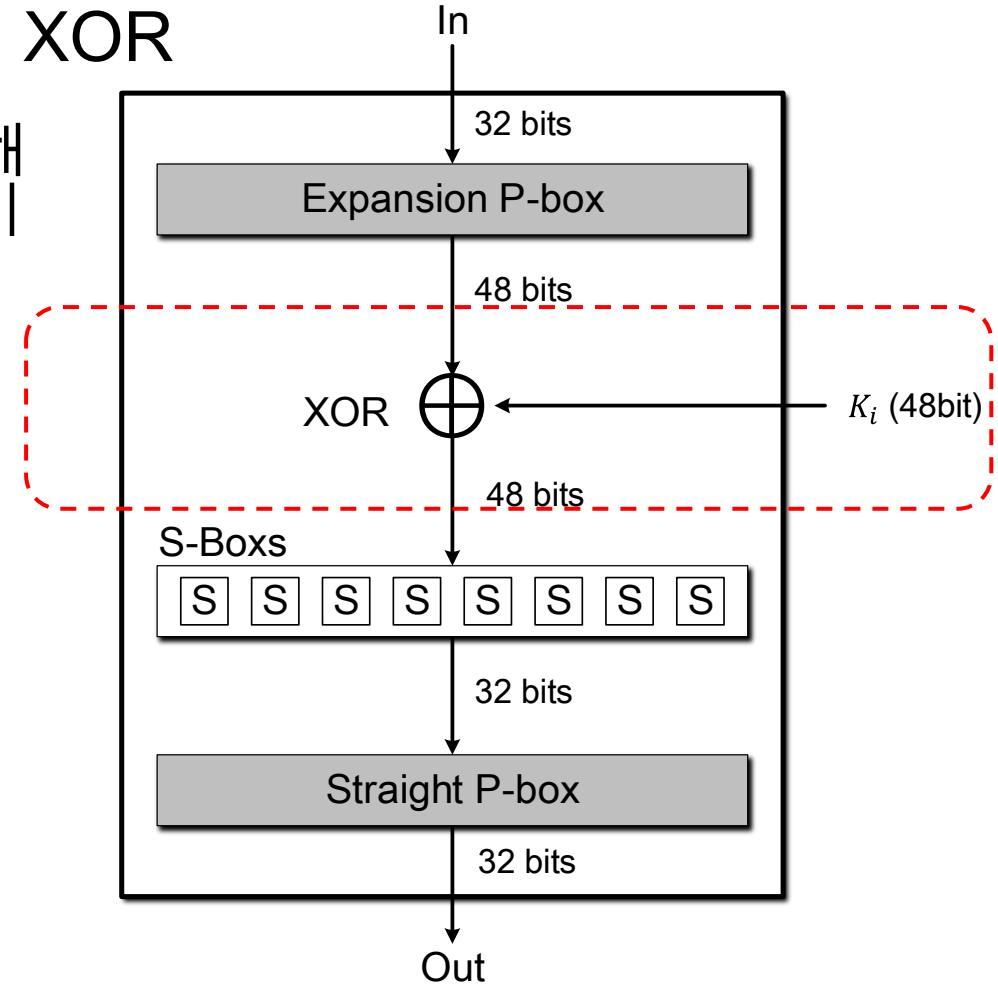
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01





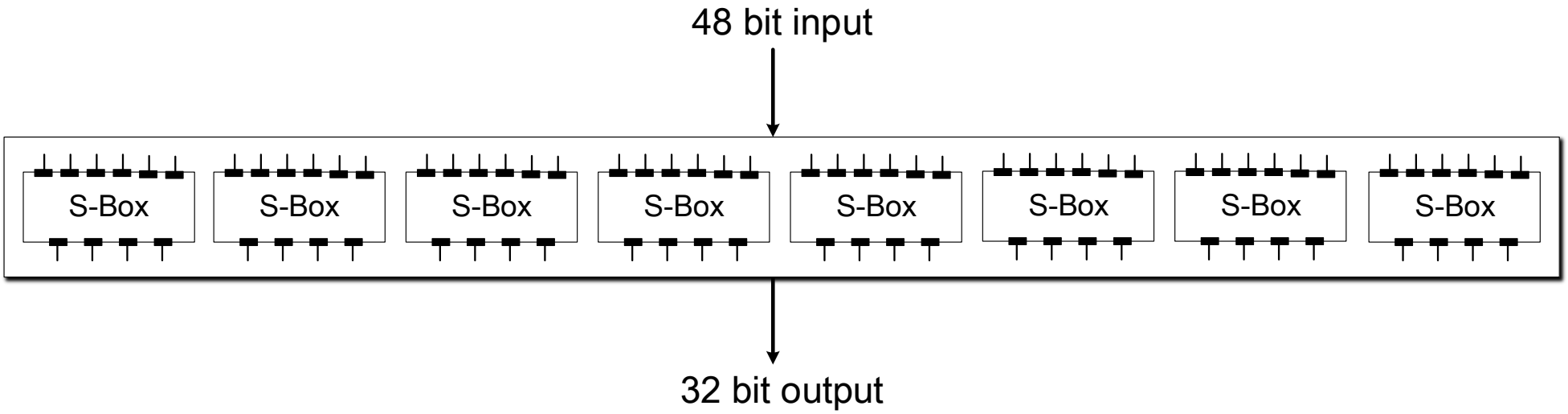
DES

- 구조
  - DES 함수 (4/7)
    - (2단계) 확장된 R_{i-1}와 K_i 를 XOR
      - K_i는 키 생성 알고리즘에 의해 만들어진 48비트의 라운드 키



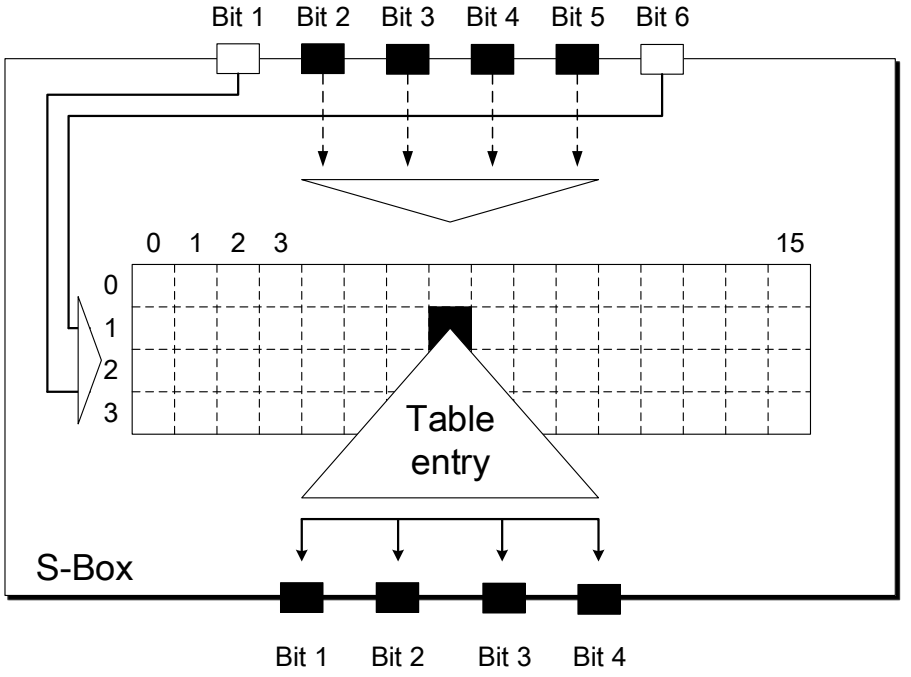
DES

- 구조
  - DES 함수 (5/7)
    - (3단계) 2단계로부터 얻은 48비트를 8부분으로 나누어 S-박스에 넣음
      - S-박스(Substitution/Choice)
        - 48비트를 다시 32비트로 줄이는 역할
        - 48비트를 6비트씩 쪼개어 8개의 부분으로 만들어 각 S-박스에 넣음
        - 각 S-박스의 결과는 4비트 값을 도출



DES

- 구조
  - DES 함수 (6/7)
    - (3단계) 2단계로부터 얻은 48비트를 8부분으로 나누어 S-박스에 넣음
      - S-박스(Substitution/Choice)
        - 각 박스마다 4행 16열을 갖는 표 기반으로 미리 결정된 규칙을 따름
        - 입력 값 비트 1과 6은 행을, 비트 2에서 5까지는 열을 결정
        - 8개의 S-박스들이 출력 값을 정의하기 위해 8개의 표가 필요



- DES 함수
  - 예제 6.3

S-박스 1의 입력 값이 100011이다. 출력 값은 무엇인가?

- 첫 번째와 여섯 번째 비트를 합치면 2진수로 11(1000**11**)이며, 10진수로 3
- 남아있는 비트는 2진수로 0001(1**0001**1)이며, 10진수로 1
- S-박스 1의 3행 1열의 값을 구하면, 그 결과 10진수로 12이고 2진수로 1100

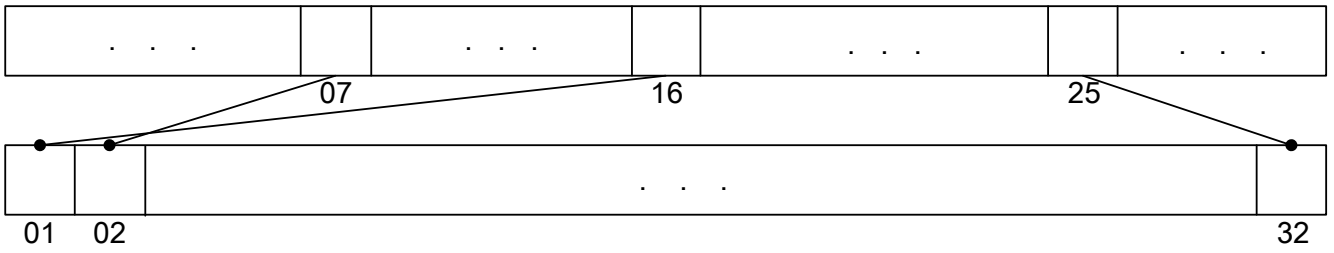
∴ 1100

	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

S-박스 1의 표

DES

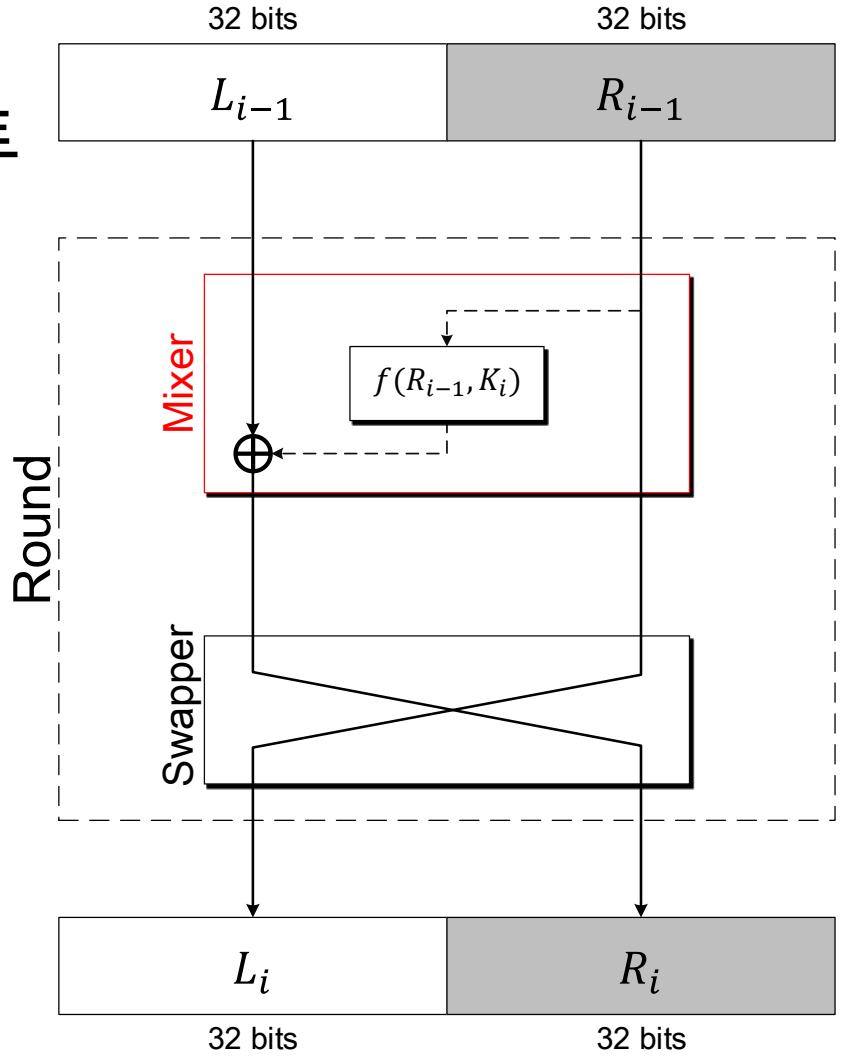
- 구조
  - DES 함수 (7/7)
    - (4단계) 3단계로부터 얻은 32비트를 단순 P-박스에 넣어 전치
      - 단순 P-박스(P table, Permutation)
        - S-박스를 통과한 32비트를 전치 시키는 역할
          - e.g., 16번째 비트가 1번째 비트로
            17번째 비트가 8번째 비트로



P table							
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

DES

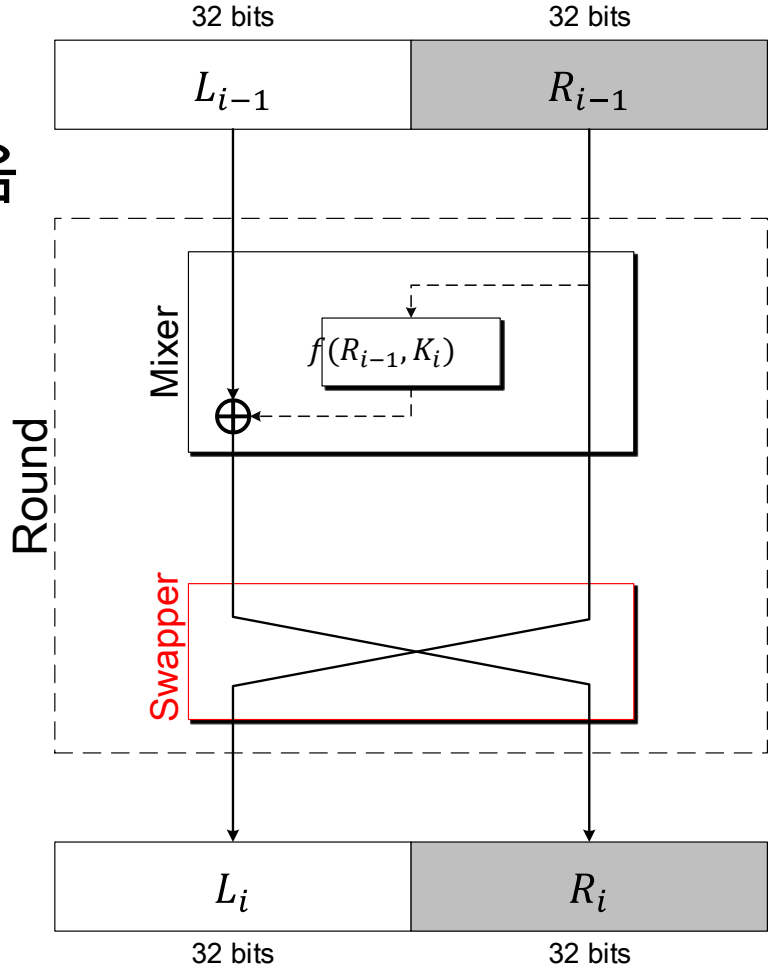
- 구조
  - 혼합기(Mixer)
    - DES함수에 의해 도출된 32비트 출력 값과 왼쪽 32비트 값을 XOR 연산함
      - $L_{i-1} \oplus f(R_{i-1}, K_i)$



DES

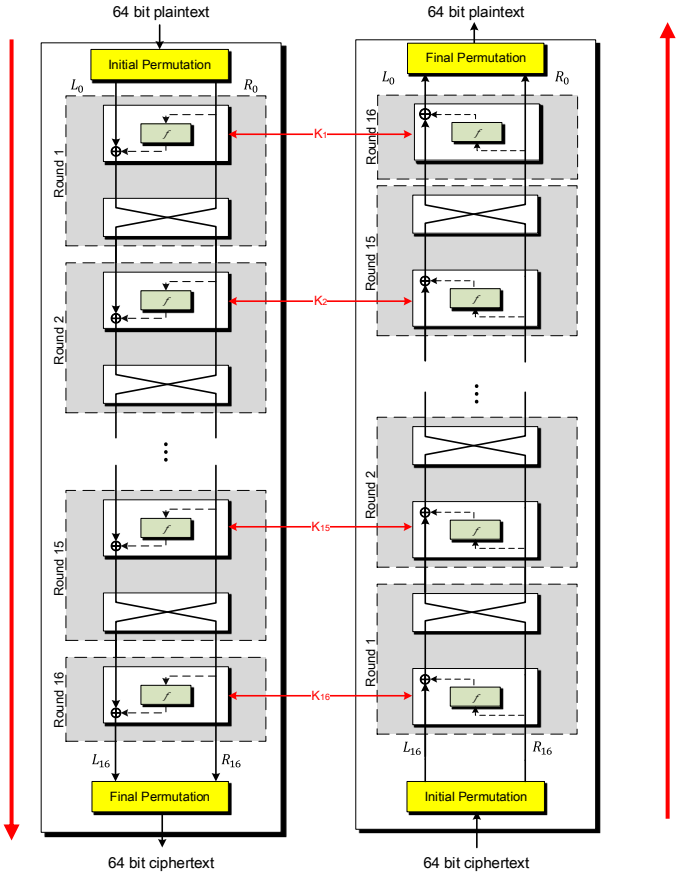
- 구조
  - 스와퍼(Swapper)
    - 혼합기를 거친 L값과 R값을 교환
    - 단, 마지막 라운드는 스와퍼가 없음

$$L_0 \qquad R_0$$
$$L_1 = R_0 \qquad R_1 = L_0 \oplus f(R_0, K_1)$$
$$L_2 = R_1 \qquad R_2 = L_1 \oplus f(R_1, K_2)$$
$$L_3 = R_2 \qquad R_3 = L_2 \oplus f(R_2, K_3)$$
$$\vdots$$
$$L_{15} = R_{14} \qquad R_{15} = L_{14} \oplus f(R_{14}, K_{15})$$
$$L_{16} = L_{15} \oplus f(R_{15}, K_{16}) \qquad L_{16} = R_{15}$$



DES

- 구조
  - DES 복호 알고리즘 (1/2)
    - 암복호 알고리즘은 서로 역함수 관계
      - 라운드의 반복과 초기 치환과 최종 치환의 역 관계성
      - 단, 라운드 키(K_i) 사용 순서를 반대로 수행
      - 일반화
        - $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ (단, 마지막 라운드 제외)



부록 참조

DES

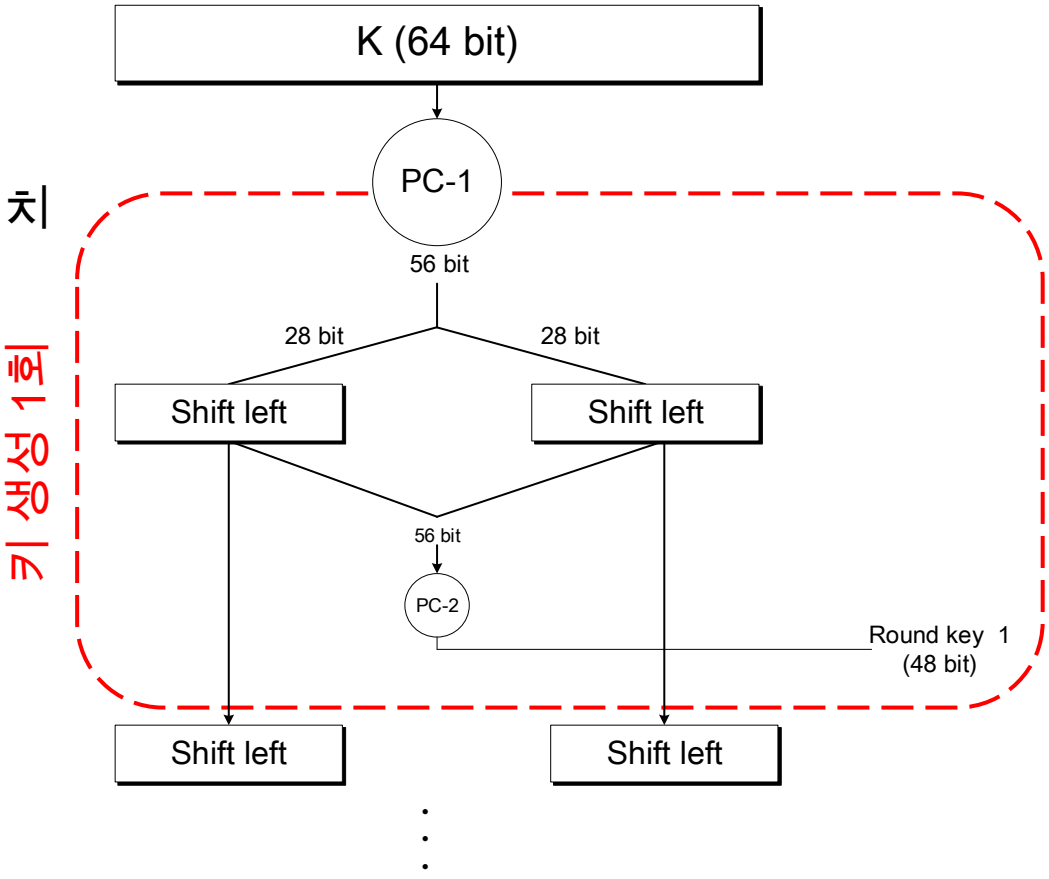
- 구조
  - DES 복호 알고리즘 (2/2)
    - 역 관계 증명
      - $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
        - $(0 \le i \le 15)$
          LE_i와 RE_i는 암호 과정의 각 왼쪽과 오른쪽 32비트
          LD_i와 RD_i는 복호 과정의 각 왼쪽과 오른쪽 32비트

① 암호화의 마지막 왼쪽, 오른쪽 블록은 복호화의 첫 번째 오른쪽, 왼쪽 블록임
- $LE_{16} = RE_{15} = RD_0,\ RE_{16} = LE_{15} \oplus f(RE_{15}, K_{16}) = LD_0$

② 일반식 적용
- $LD_1 = RD_0 = RE_{16} = RE_{15}$(과정①)
  $RD_1 = LD_0 \oplus f(RD_0, K_{16})$

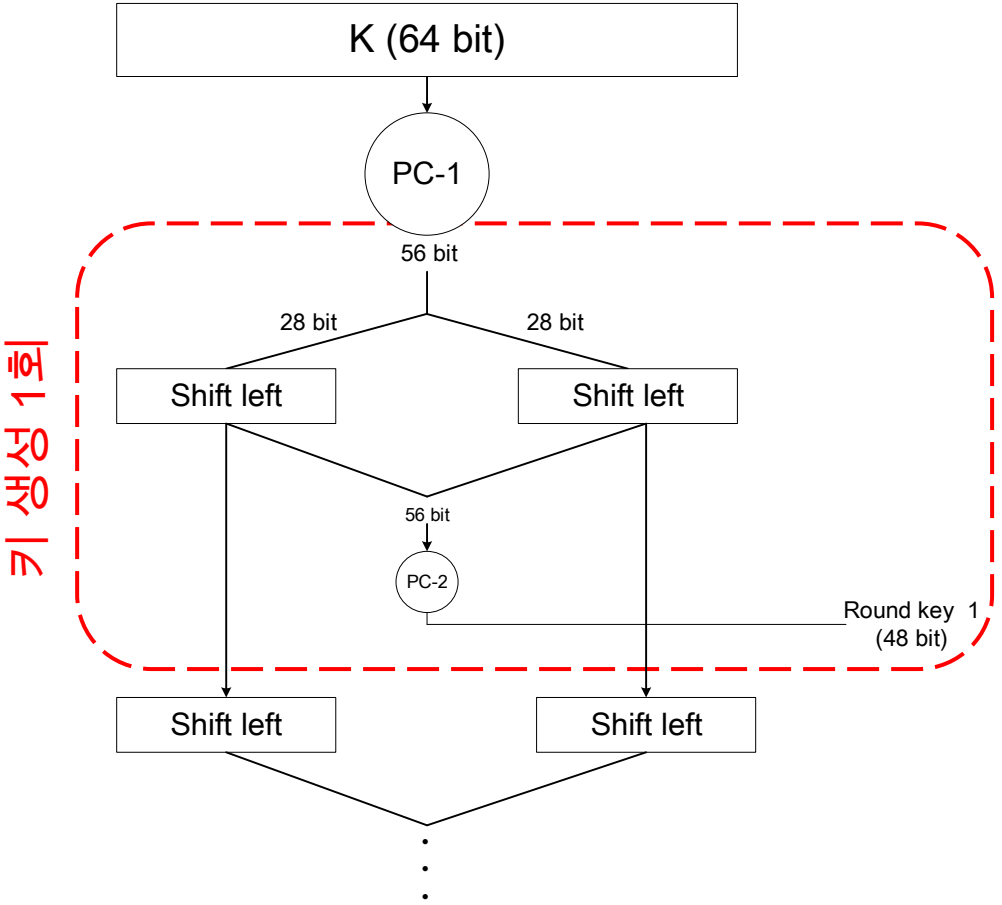
DES

- 구조
  - 라운드 키 생성 (1/7)
    - 64비트의 암호 키로부터 16개의 48비트 라운드 키를 생성
    - 단계
      - (1단계)
        - PC-1을 적용해 56비트로 전치
      - (2단계)
        - 28비트씩 두 블록으로 나누고 좌측 순환 이동



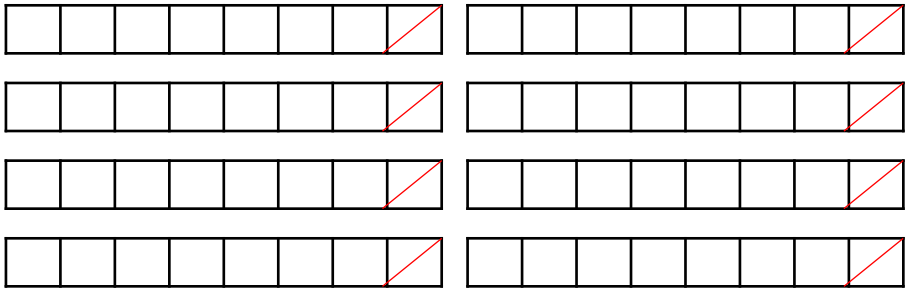
DES

- 구조
  - 라운드 키 생성 (2/7)
    - 64비트의 암호 키로부터 16개의 48비트 라운드 키를 생성
    - 단계
      - (3단계)
        - 3단계 값들을 다시 합하여 PC-2를 적용해 56비트를 48비트로 전치
      - (4단계)
        - 2단계로 돌아가 반복



DES

- 구조
  - 라운드 키 생성 (3/7)
    - (1단계) PC-1(Permutation Choice Table)적용해 56비트로 전치
      - PC-1을 통해 키와 패러티 비트를 가지고 있던 64비트는 Autual key인 56비트로 전치됨
      - PC-1은 축소 P-박스의 일종으로, 8의 배수가 버려진다는 특징이 있음
      - e.g., 57번째 비트는 1번째로 전치
        4번째는 56번째로 전치



PC-1							
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

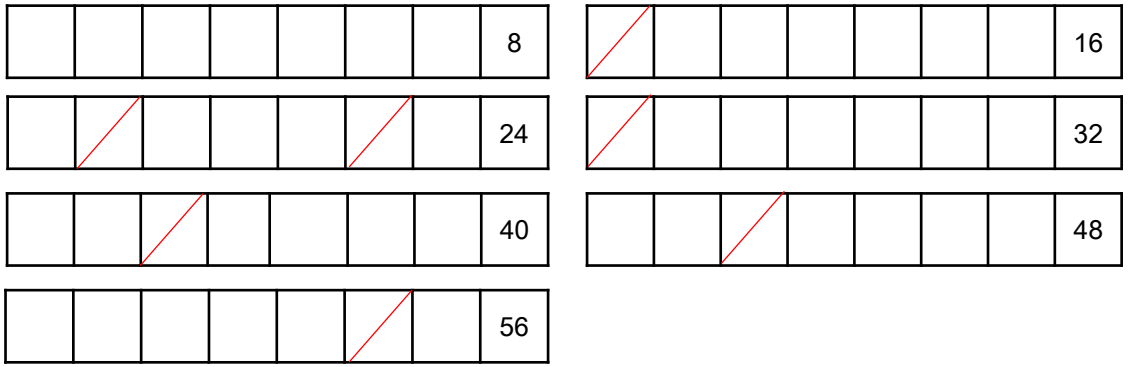
DES

- 구조
  - 라운드 키 생성 (4/7)
    - (2단계) 전치한 56비트를 28비트씩 나누고 좌측 순환 이동
      - 각 부분의 왼쪽으로 1 또는 2비트씩 순환 이동
      - 각 라운드마다 다르게 이동
        - e.g., 1라운드: 10011010 → 00110101
          7라운드: 10011010 → 01101010

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

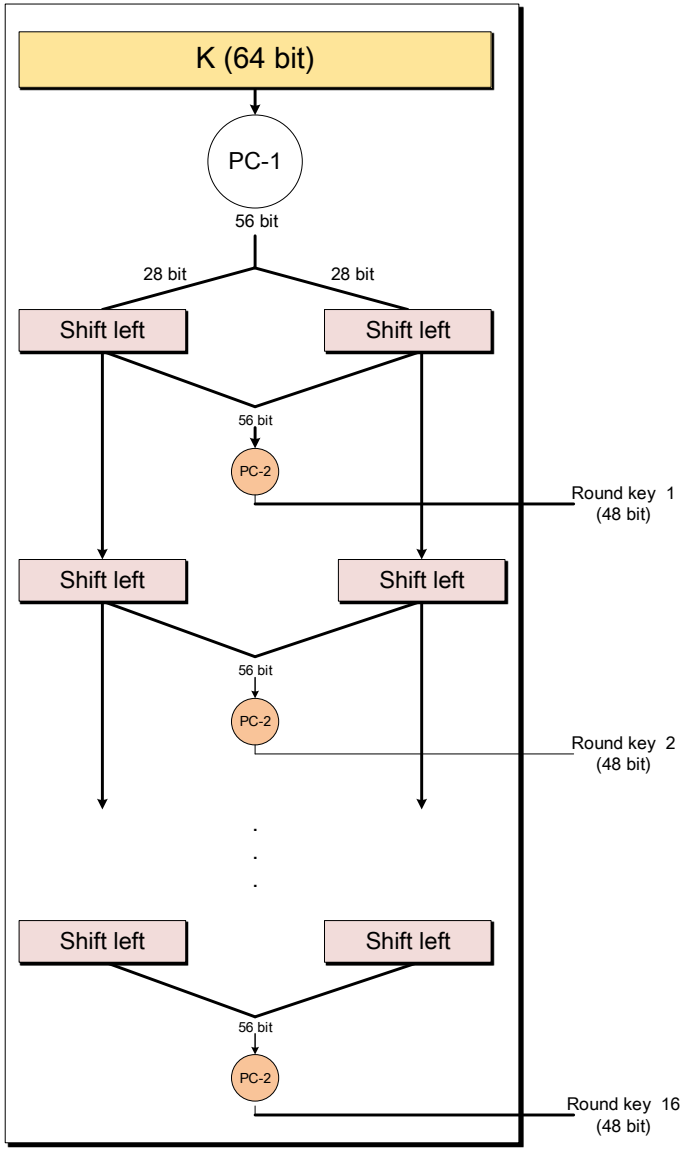
- 구조
  - 라운드 키 생성 (5/7)
    - (3단계) PC-2 테이블(Compression Table) 적용
      - 2단계로부터 온 두 블록을 합쳐서 56비트를 만들고 PC-2를 적용시켜 48비트로 축약 전치
      - 9, 18, 22, 25, 35, 43, 54번째 비트 제거
      - e.g., 14번째 비트를 1번째로 전치
        32번째 비트를 48번째로 전치



PC-2							
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

DES

- 구조
  - 라운드 키 생성 (7/7)
    - (4단계) 2단계부터 반복
      - 16개의 라운드 키가 생성될 때까지 반복



DES

- 분석
  - 페이스텔 구조 사용
    - 전치(Permutation)와 대치(Substitution)를 번갈아 수행하여 혼돈과 확산의 성질을 만족함
    - 강한 쇄도 효과(Avalanche Effect)를 가짐
      - 암호 알고리즘이 입력 값에 미세한 변화를 줄 경우 출력 값에 상당한 변화가 일어나는 성질
    - 완전성(Completeness) 효과를 가짐
      - 암호화 과정에서 모든 비트가 다른 모든 비트에 영향을 미치는 속성
    - 암·복호화 알고리즘이 동일한 구성요소를 사용

DES

- 성질 – 쇄도 효과 (1/2)
  - 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.

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

  평문: 0000000000000001 키: 22234512997ABB23
  암호문: 0A4ED5C15A63FEA3

- 한 비트만 다를지라도, 암호문은 29비트가 다름. 평문의 약 1.5%가 변하는 것은 암호문의 약 45%의 변화를 나타낼 수 있음을 의미

DES

- 성질 – 쇄도 효과 (2/2)
  - 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

예제 6.7에 대한 비트 차이의 개수

DES

- 분석
  - 설계 기준
    - S-박스
      - 각 라운드로부터 다음 라운드까지 혼돈과 확산 성질을 만족하도록 설계됨
      - 비선형 방식으로 설계함
        - S-박스가 없으면 단순 XOR 연산이라 DES 함수가 선형함수가 됨
      - 입력 값의 한 비트를 바꾼다면, 출력 값에서 두 비트 이상이 바뀜
      - 두 입력값이 첫 번째 두 개의 비트(비트 1, 2)가 다르고 마지막 두 개의 비트(비트 5, 6)가 같다면 두 출력 값이 달라야 함
        - e.g., 8번째 S-박스에서 011011→14, 10011→07(부록 참조)

DES

- 분석
  - 설계 기준
    - P-박스
      - 32비트 입력에서 32비트 출력으로 가는 하나의 단순 P-박스와 32비트의 입력에서는 48비트의 출력으로 가는 하나의 확장 P-박스로 구성됨
      - 한 S-박스로부터 생성된 출력 비트들 중 어떤 2개의 출력도 다음 라운드의 동일한 S-박스로 가지 않음

        (이때, S_i는 i번째 S-박스를 의미하며 $2 \leq j \leq 7$)
      - S_{j-2}의 한 출력 비트는 다음 라운드에서 S_j의 첫 번째 비트들로 감
      - S_{j-1}의 한 출력 비트는 다음 라운드에서 S_j의 마지막 비트들로 감
      - S_{j+1}의 한 출력 비트는 다음 라운드에서 S_j의 가운데 비트들 중 하나로 감

DES

- 분석
  - 알고리즘 설계 취약점
    - S-박스(1/3)
      - 4번째 S-박스의 4비트 출력 중에서 하위 3개의 출력 비트들은 입력 비트의 일부의 보수를 취함으로써 첫 번째 출력 비트와 동일한 방법으로 얻을 수 있음
        - 입력 $X = x_1\ x_2\ x_3\ x_4\ x_5\ x_6$, 출력 $Y = y_1 y_2\ y_3\ y_4$일 때,
          $X = 101011$일 때, $y_2\ y_3\ y_4 = 010$
          이때, 010의 일부를 보수 취하면 y_1
          (e.g., 101, 000, 001)

DES

- 분석
  - 알고리즘 설계 취약점
    - S-박스(2/3)
      - 하나의 S-박스에 입력된 서로 다른 두 값은 동일한 값으로 출력될 가능성이 있음
        - 입력 값과 출력 값 사이의 단순한 1:1대응이 아니다.
        - e.g., 출력 값 1110 0100 1010을 가지기 위해서는 입력 값이 0000 1010 011110뿐만 아니라 111111 110011 100001도 가능

S1	
000000	1110
111111	1110

S2	
101010	0100
110011	0100

S3	
011110	1010
100001	1010

DES

- 분석
  - 알고리즘 설계 취약점
    - S-박스(3/3)
      - 세 개의 이웃하는 S-박스들에 들어가는 입력 비트를 바꾸는 것만으로도 하나의 라운드에 대해 동일한 출력 값을 얻는 것이 가능
        - 라운드 N번째의 입력 값: 000000 101010 011110 → 000000 101011 011110
          라운드 N+i번째의 입력 값: 111111 110011 100001
          라운드 N번째에서 입력 비트를 바꾸면, 라운드 N+i번째의 출력 값과 라운드 N번째 출력 값이 동일함

S1	
000001	1000
111111	1110
000000	1110

S2	
101010	0000
110011	0100
101011	0100

S3	
011110	1010
100001	1010

DES

- 분석
  - 알고리즘 설계 취약점
    - P-박스
      - 설계자들이 왜 초기 전치와 최종 전치를 사용했는지 명백하지 않음
      - 확장 치환에서 4비트로 단위로 분할했을 때, 첫 번째 비트와 네 번째 비트들은 반복됨

DES

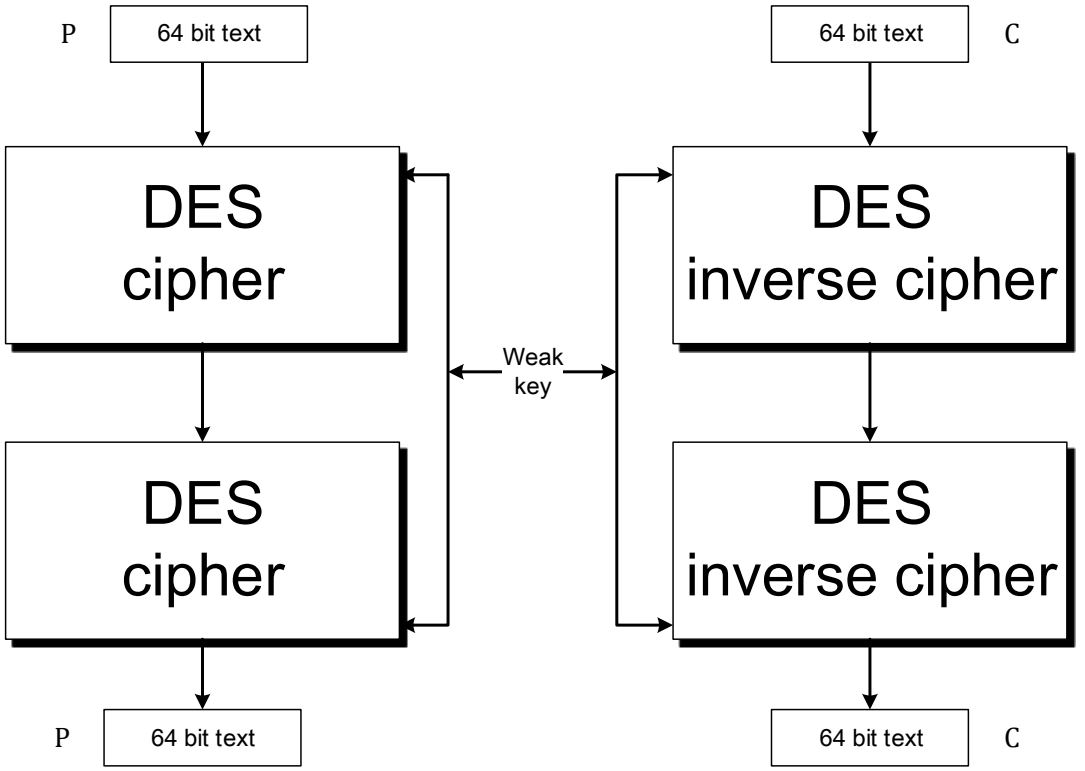
- 분석
  - 암호 키 크기
    - 현재 과학 기술로, 2^{56}의 키를 조사하는 것은 어렵지 않음
      - 1977년에 RSA 연구소의 과제로 3500개의 컴퓨터를 사용하여 120일 만에 키를 찾아냄
      - 1996년에 DES Challenge II에서 250,000달러의 전용칩을 제작해 56시간만에 암호를 해독하였음
      - 1999년 DES Challenge III에서 1만 여대의 컴퓨터와 전용칩을 이용하여 22시간 15분만에 암호를 해독하였음

DES

- 분석
  - 암호 키
    - 취약 키 (Weak Keys) (1/2)
      - 패리티 제거 연산 이후에, 모두 0이거나 모두 1이거나 절반은 0이고 절반 1을 구성하는 키
      - 취약 키로 생성된 라운드 키들은 서로 동일한 패턴을 가짐
        - 키 생성 알고리즘이 암호키를 절반씩 나누고 순환이동 또는 전치 박스를 거치는데 비트가 같을 경우엔 순한이동 또는 전치 박스를 적용해도 전체 블록 값이 변하지 않기 때문임
        - e.g., 0101 0101 0101 0101로 만들어진 16라운드 키들은 모두 0으로 되어 있음

64비트 키	56비트 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

- 분석
  - 암호 키
    - 취약 키 (Weak Keys) (2/2)
      - 취약 키를 2번 암호화하면 평문이 나옴
        - $E_k(E_k(P)) = P$ 역관계



DES

- 분석
  - 암호 키
    - 준 취약 키 (Semi Weak Keys) (1/2)
      - 두 가지 형태의 라운드 키만 생성

첫 번째 키 (A)	두 번째 키 (B)
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

- 분석
  - 암호 키
    - 준 취약 키 (Semi Weak Keys) (2/2)
      - 하나의 쌍으로 묶인 두 개의 준 취약 키는 순서만 다를 뿐 동일한 라운드 키를 생성
        - e.g.,첫 번째 키 쌍(01FE 01FE 01FE 01FE / FE01 FE01 FE01 FE01)

	(A)	(B)
Round key1	9153E54319BD	6EAC1ABCE642
Round key2	6EAC1ABCE642	9153E54319BD
Round key3	6EAC1ABCE642	9153E54319BD
Round key4	6EAC1ABCE642	9153E54319BD
Round key5	6EAC1ABCE642	9153E54319BD
Round key6	6EAC1ABCE642	9153E54319BD
Round key7	6EAC1ABCE642	9153E54319BD
Round key8	6EAC1ABCE642	9153E54319BD
Round key9	9153E54319BD	6EAC1ABCE642
Round key10	9153E54319BD	6EAC1ABCE642
Round key11	9153E54319BD	6EAC1ABCE642
Round key12	9153E54319BD	6EAC1ABCE642
Round key13	9153E54319BD	6EAC1ABCE642
Round key14	9153E54319BD	6EAC1ABCE642
Round key15	9153E54319BD	6EAC1ABCE642
Round key16	6EAC1ABCE642	9153E54319BD

DES

- 분석
  - 암호 키
    - 잠재적 취약 키(Possible Weak Keys)
      - 4개의 라운드 키만 생성해내는 키
        - 16라운드를 다 합하여도 4개만 나옴

1F 1F 01 01 0E 0E 01 01	FE E0 1F 01 FE F1 0E 01	FE 1F E0 01 FE 0E F1 01	01 01 FE FE 01 01 FE FE
E0 01 01 E0 F1 01 01 F1	E0 01 1F FE F1 01 0E FE	E0 1F FE 01 F1 0E FE 01	01 01 E0 E0 01 01 F1 F1
01 1F 1F 01 01 0E 0E 01	E0 FE 1F 01 F1 FE 0E 01	1F 1F E0 E0 0E 0E F1 F1	01 FE FE 01 01 FE FE 01
FE 1F 01 E0 FE 0E 01 F1	FE 1F 1F FE FE 0E 0E FE	FE 01 E0 1F FE 01 F1 0E	1F 1F FE FE 0E 0E FE FE
1F 01 01 1F 0E 01 01 0E	FE E0 01 1F FE F1 01 0E	1F 01 FE E0 0E 01 FE F1	1F E0 E0 1F 0E F1 F1 0E
FE 01 1F E0 FE 01 0E F1	1F FE 01 E0 0E FE 01 F1	E0 01 FE 1F F1 01 FE 0E	FE FE E0 E0 FE FE F1 F1
01 01 1F 1F 01 01 0E 0E	E0 FE 01 1F F1 FE 01 0E	01 1F FE E0 01 0E FE F1	01 FE E0 1F 01 FE F1 0E
E0 1F 1F E0 F1 0E 0E F1	01 FE 1F E0 01 FE 0E F1	01 E0 E0 01 01 F1 F1 01	E0 FE FE E0 F1 FE FE F1
E0 E0 01 01 F1 F1 01 01	E0 E0 1F 1F F1 F1 0E 0E	1F 01 E0 FE 0E 01 F1 FE	01 E0 FE 1F 01 F1 FE 0E
FE 01 01 FE FE 01 01 FE	1F E0 01 FE 0E F1 01 FE	1F FE E0 01 0E FE F0 01	FE E0 E0 FE FE F1 F1 FE
FE FE 01 01 FE FE 01 01	FE FE 1F 1F FE FE 0E 0E	01 1F E0 FE 01 0E F1 FE	1F FE FE 1F 0E FE FE 0E
E0 1F 01 FE F1 0E 01 FE	01 E0 1F FE 01 F1 0E FE	1F E0 FE 01 0E F1 FE 01	E0 E0 FE FE F1 F1 FE FE

DES

- 분석
  - 암호 키
    - 키 보수
      - 비트는 0과 1로 표현됨
        - 즉, 보수들로 표현이 되는 것
      - 암호 해독자는 가능한 모든 키를 시도(2^{56})하지 않고 절반만 (2^{55}) 검사하고, 나머지 키에 대해서는 보수의 성질을 이용하여 값을 찾음
        - 키와 평문에 모두 보수를 취하여 암호화하면, 원래의 암호문의 보수를 얻을 수 있음
          - e.g., $C = E(K, P) \rightarrow \bar{C} = E(\bar{K}, \bar{P})$

	Original	Complement
K	1234123412341234	EDCBEDCBEDCBEDCB
P	12345678ABCDEF12	EDCBA987543210ED
C	E112BE1DEFC7A367	1EED41E210385C98

DES

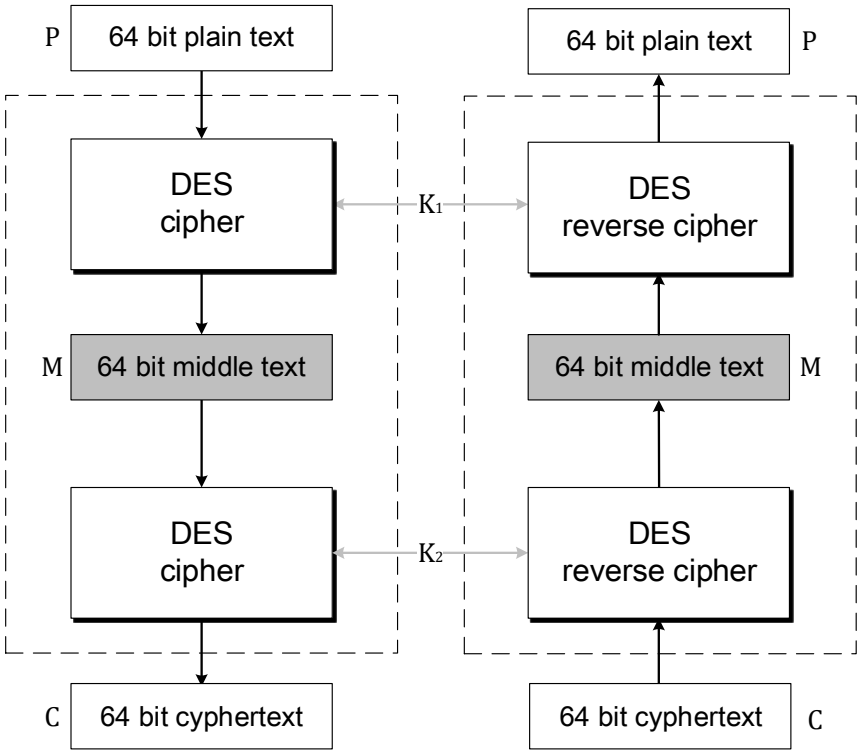
- 분석
  - 안전성 (1/2)
    - 전수 조사 공격
      - 단일 DES를 짧은 키 길이와 보수 성질로 암호가 해독되어 현재는 사용하지 않으나, 이를 보완하여 다중 DES를 사용
        - 다중 DES는 최소 2개의 키와 키 길이가 최소 112비트를 사용하여 보완
    - 선형공격
      - 1993년 일본 암호학자 마츠이가 소개한 방법으로, 암호문과 평문 사이의 선형 관계를 이용하여 키를 찾는 방법
        - 입력 값과 출력 값의 상관관계성을 해석
        - 2^{43}개의 선택 평문 쌍을 이용해 56비트 키를 찾을 수 있음을 입증
        - IBM 3090 컴퓨터를 사용하여 DES를 50시간만에 해독한 사례가 있음

DES

- 분석
  - 안전성 (2/2)
    - 차분 공격
      - 1990년대 Eli Biham과 Adi Shamir가 소개한 방법으로, 평문 쌍과 차이와 암호문 쌍의 차이를 분석하여 암호 키를 찾는 방법
        - 2^{47}개의 선택 평문 쌍을 이용해 56비트 키를 찾을 수 있음을 입증함

DES

- 다중 DES
  - 이중 DES(2DES, Double DES)
    - DES 과정을 두 번 반복한 알고리즘으로, 키 공간이 112 비트로 증가
      - 실질 탐색 횟수는 2^{57}번

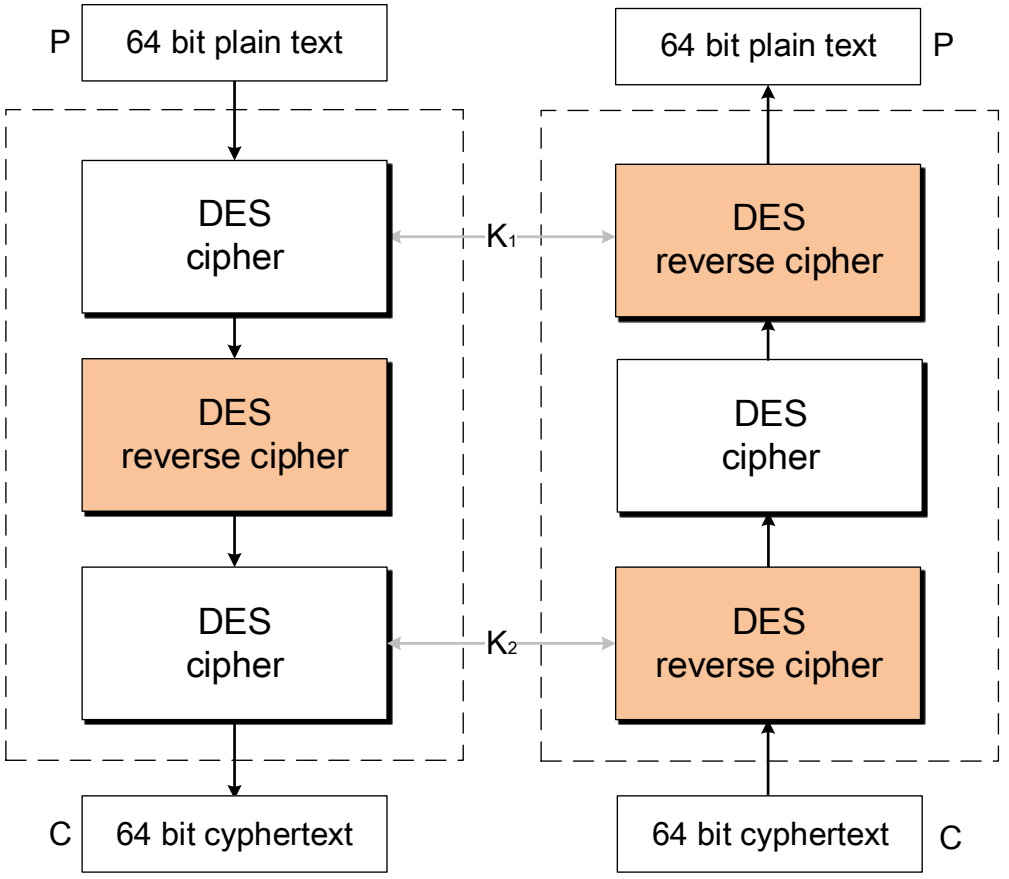


DES

- 다중 DES
  - 이중 DES(2DES, Double DES)
    - 중간 일치 공격에 취약함
      - $\mathrm{E}_{\mathrm{K}_1}(\mathrm{P}) = \mathrm{M} = \mathrm{D}_{\mathrm{K}_2}(\mathrm{C})$
      - 공격자가 P, C를 안다고 했을 때, K_1의 모든 가능한 값(2^{56})을사용하여 P를 암호화하고 M에 대한 테이블을 만들고, K_2의 모든 가능한 값(2^{56})을 사용하여 C를 복호화하고 M에 대한 테이블을 만듦
      - M이 같은 K_1, K_2쌍이 나올 때까지 후보 키 쌍을 검사함
        - 즉, 2^{112}검사가 아니라 2×2^{56}검사를 하면 됨

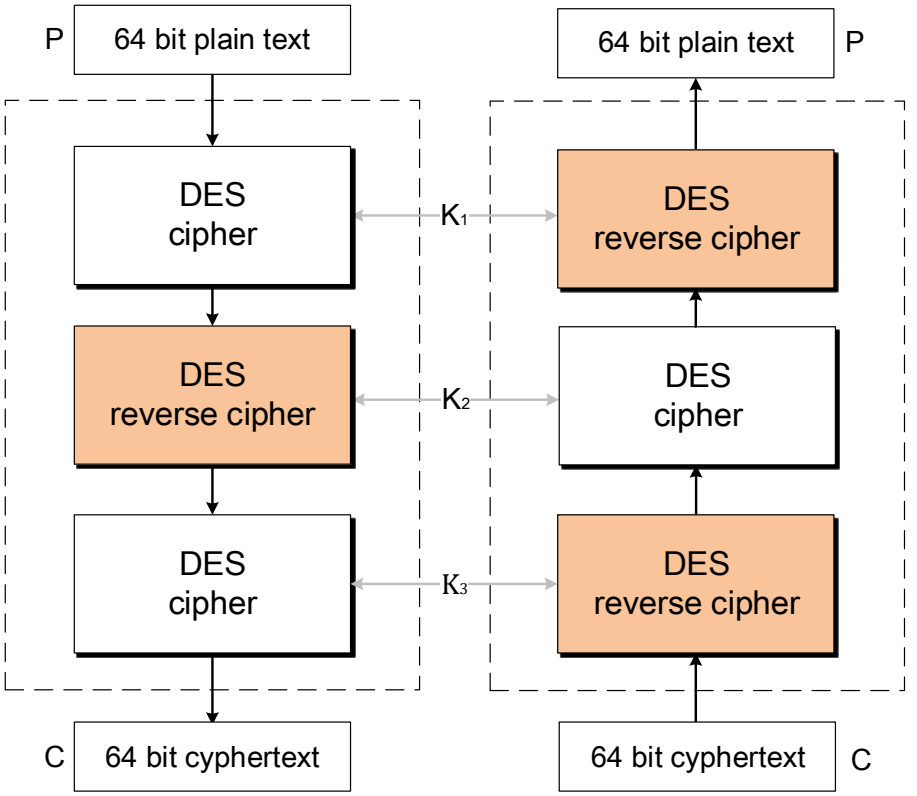
DES

- 다중 DES
  - 삼중 DES(3DES, Triple DES)
    - 두 개의 키를 갖는 삼중 DES
      - 중간 단계에 복호화
        - 이중 DES의 취약점 보완
        - $k_1 = k_2 = k$라고 해도, 이중 DES보다는 복잡하기에 더 강하다고 할 수 있음



DES

- 다중 DES
  - 삼중 DES(3DES, Triple DES)
    - 세 개의 키를 갖는 삼중 DES
      - K_1로 암호화하고, K_2로 복호화하고, K_3로 암호화하며 복호화 시 반대로 수행



목 차

AES(Advanced Encryption Standard)

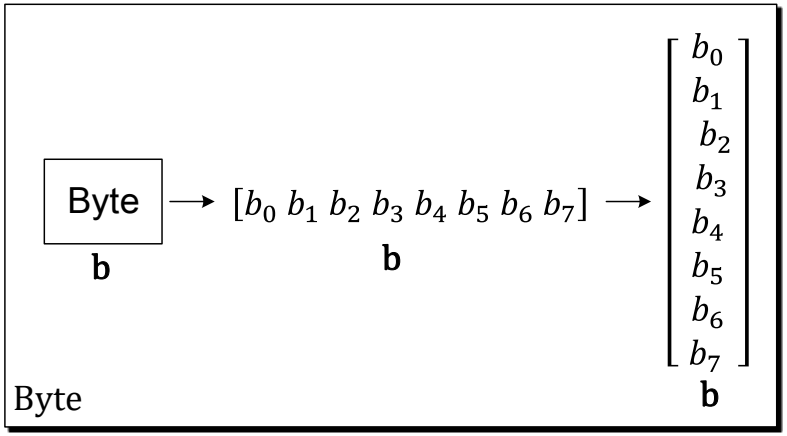
- 개요
  - 정의
    - 128비트의 평문을 128비트의 암호문으로 암호화하는 대칭키 블록 암호
    - AES는 세 가지 키 길이를 지원
      - AES-128
        - 128비트 키(16바이트), 10라운드
      - AES-192
        - 192비트 키(24바이트), 12라운드
      - AES-256
        - 256비트 키(32바이트), 14라운드

AES

- 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
        - Rijdael 알고리즘이 선정 기준 세 가지 조건들(안전성, 비용, 구현)에 가장 부합
    - 2001년
      - Rijdael 알고리즘을 기반으로한 AES를 FIPS 197로 공식 발표

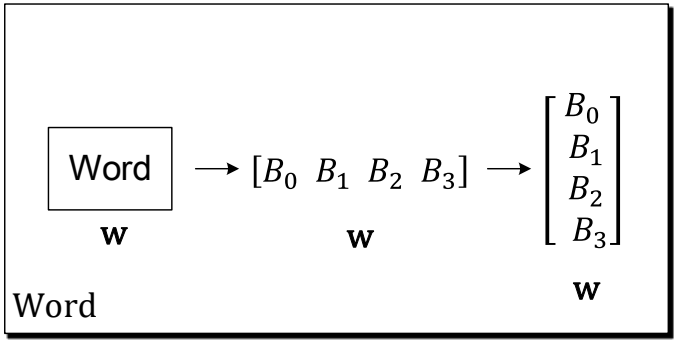
AES

- 데이터 단위 (1/3)
  - 비트(Bit)
    - 2진 데이터 값으로, 더 이상 쪼개질 수 없는 가장 작은 단위
    - 표현 시, 소문자로 표현
      - e.g., b_1, b_2
  - 바이트(Byte)
    - 한 바이트당 8개의 비트로 이루어진 단위
    - 비트를 원소로 하는 행 행렬(1×8) 또는 열 행렬(8×1)로 표현
    - 표현 시, 볼드체 소문자로 표현
      - e.g., $\mathbf{b}$

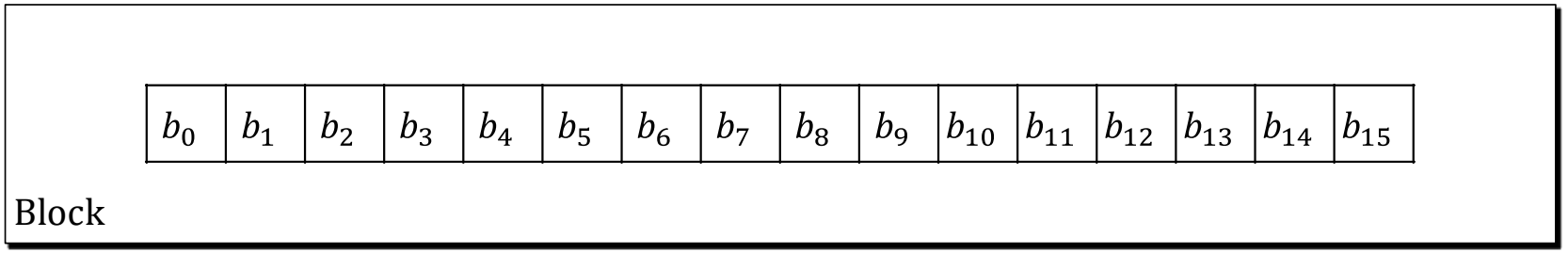


AES

- 데이터 단위 (2/3)
  - 워드(Word)
    - 한 워드당 4바이트로 이루어진 단위
    - 바이트를 원소로 하는 행 행렬(1×4) 또는 열 행렬(4×1)로 표현
    - 표현 시, 볼드체의 소문자 w 사용
      - e.g., **w**



  - 블록(Block)
    - 128비트(16바이트)로 이루어진 단위
    - 16바이트를 원소로 하는 행 행렬(1×16)로 표현



AES

- 데이터 단위 (3/3)
  - 상태(State)
    - 라운드는 여러 단계로 구성되는데 각 단계 전후에 있는 블록을 말함
    - 16바이트를 원소로 가지는 4×4 행렬로 표현(각 원소는 $\mathbf{S_{r,c}}$로 표현)
      - $\mathbf{r}$은 행, $\mathbf{c}$는 열
    - 표현 시, S라 나타내며 임시 상태일 시에는 T로 표현

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

State

AES

- 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 얻는다

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 표현한다

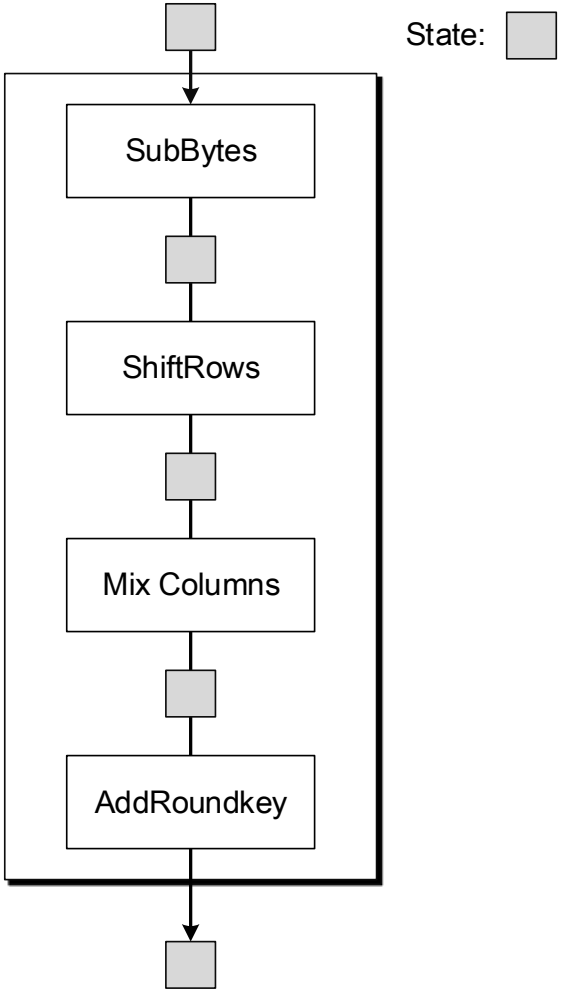
00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19

- 이를 왼쪽 문자열부터 한 열로 채운다

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

AES

- 라운드(Round)
  - AES의 라운드 함수는 4가지 형태의 변환을 사용
    - 변환
      - 대치(Substitution)
      - 치환(Permutation)
      - 뒤섞음(Mixing)
      - 키 덧셈(Key-Adding)

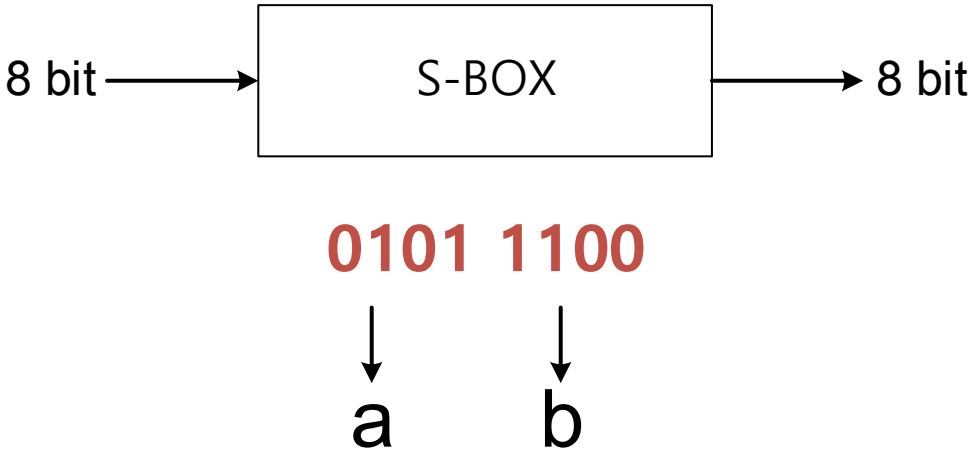


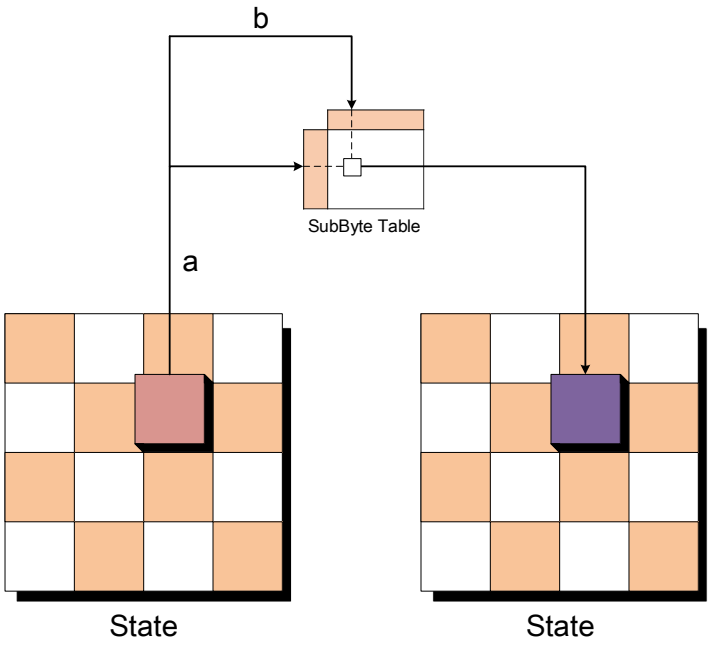
AES

- 라운드(Round)
  - 대치(Substitution) (1/5)
    - 부분바이트 (SubBytes)
      - AES의 암호과정에 사용되는 대치 함수
      - 부분바이트 연산은 16개의 독립된 바이트 단위의 변환을 수행
        - 상태배열의 모든 16개 바이트에 대해 독립적으로 수행함을 의미
      - 대치 변환은 두 가지 방법으로 정의가 됨
        - 표 참조 과정
        - 체 $GF(2^8)$를 이용한 변환 방법
    - 역서브바이트(InSubBytes)
      - SubBytes의 역 변환

AES

- 라운드(Round)
  - 대치(Substitution) (2/5)
    - 부분바이트 (SubBytes)
      - 표 참조 과정
        - 하나의 바이트를 대치하기 위해 각 바이트를 4비트씩 2개의 16진수로 계산하여 왼쪽 4비트를 S-박스의 행으로, 오른쪽 4비트를 열로 표를 읽음
          - 즉, 16진수의 행과 열이 교차하는 부분의 바이트 값을 출력함





- 라운드(Round)
  - 대치(Substitution) (3/5)
    - 부분바이트 (SubBytes)
      - 표 참조 과정
        - e.g., 아래와 같은 State가 있을 때, $S_{0,0}$인 0x19에서 상위 4비트 1은 행으로, 하위 4비트 9는 열로 읽음 (부록 참고)

$$\begin{bmatrix} \mathbf{S_{0,0}} & \mathbf{S_{0,1}} & \mathbf{S_{0,2}} & \mathbf{S_{0,3}} \\ \mathbf{S_{1,0}} & \mathbf{S_{1,1}} & \mathbf{S_{1,2}} & \mathbf{S_{1,3}} \\ \mathbf{S_{2,0}} & \mathbf{S_{2,1}} & \mathbf{S_{2,2}} & \mathbf{S_{2,3}} \\ \mathbf{S_{3,0}} & \mathbf{S_{3,1}} & \mathbf{S_{3,2}} & \mathbf{S_{3,3}} \end{bmatrix} = \begin{bmatrix} 0x19 & 0xa0 & 0x9a & 0xe9 \\ 0x3d & 0xf4 & 0xc6 & 0xf8 \\ 0xe3 & 0xe2 & 0x8d & 0x48 \\ 0xbe & 0x2b & 0x2a & 0x08 \end{bmatrix} = \begin{bmatrix} 0xd4 & 0x8e & 0x4d & 0xa1 \\ 0x3d & 0x9d & 0x92 & 0x4e \\ 0x0b & 0xea & 0x0c & 0x4c \\ 0x70 & 0x59 & 0x1a & 0x0e \end{bmatrix}$$

AES

- 라운드(Round)
  - 대치(Substitution) (4/5)
    - 부분바이트 (SubBytes)
      - 체 $GF(2^8)$를 이용한 변환 방법
        - 기약 다항식 $x^8 + x^4 + x^3 + 1$를 가진 체 $GF(2^8)$를 이용하여 대수적인 변환으로 S-박스를 정의
        - SubByte: $\mathrm{d} = \mathrm{X}(\mathrm{S}_{\mathrm{r,c}})^{-1} \oplus \mathrm{y}$
          InSubByte: $[\mathrm{X}^{-1}(\mathrm{d} \oplus \mathrm{y})]^{-1} = [\mathrm{X}^{-1}(\mathrm{X}(\mathrm{S}_{\mathrm{r,c}})^{-1} \oplus \mathrm{y} \oplus \mathrm{y})]^{-1} = (\mathrm{S}_{\mathrm{r,c}})^{-1}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

고정 행렬 X

AES

- 부분바이트(SubBytes)
  - 예제 7.3

16진수 0C가 SubByte 통해 FE로 계산되는 과정을 보이고 역으로 InSubByte를 통해 다시 0C로 계산하는 과정을 설명하라.

$$d = X(S_{r,c})^{-1} \oplus y$$

- $GF(2^8)$ 체에서 곱셈에 대한 역원을 구함
  ∴ B0
- 이를 비트 단위로 표현 하면 B0 = $(10110000)_2$
- $(10110000)_2$값을 최하위 비트를 위로가게 한 열 행렬로 변환
- 이를 고정된 값인 정방행렬 X과 곱셈연산
  $(10110000)_2 \rightarrow (10011101)_2$
- 이를 고정된 값인 열 행렬 y와 XOR 연산함
  $(10011101)_2 \rightarrow (1111110)_2 = d$

- 부분바이트(SubBytes)
  - 예제 7.3

16진수 0C가 SubByte 통해 FE로 계산되는 과정을 보이고 역으로 InSubByte를 통해 다시 0C로 계산하는 과정을 설명하라.

$$[X^{-1}(d \oplus y)]^{-1} = [X^{-1}(X(S_{r,c})^{-1} \oplus y \oplus y)]^{-1} = (S_{r,c})^{-1}$$

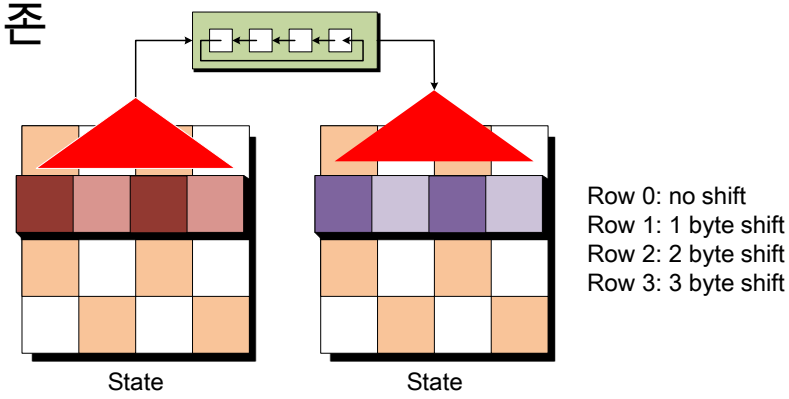
- XOR연산을 수행
  $d \oplus y = (10011101)_2$
- X^{-1}을 곱함
  $[X^{-1}(d \oplus y)] = B0 = (10110000)_2$
- B0 = $(10110000)_2$의 곱셈에 대한 역원은 0C

AES

- 라운드(Round)
  - 대치(Substitution) (5/5)
    - 부분바이트 (SubBytes)
      - SubBytes연산은 비선형 연산임
        - SubBytes 단계에서 각 바이트를 변환할 때 작업 1과 작업 2가 이루어지는데 이 두 가지 작업을 결합하는 것은 최종 비선형 결과를 만들어냄
          - 작업 1: $GF(2^8)$ 유한체에서 각 원소에 대해 곱셈에 대한 역원을 찾는 연산은 비선형 연산
          - 작업 2: 변환된 값을 특정한 행렬과 곱하고 일정한 값을 더하는 연산은 선형 연산

AES

- 라운드(Round)
  - 치환(Permutation) 또는 이동 연산(Shifting)
    - ShiftRows
      - 암호화 과정에서 사용
      - 바이트 단위로 왼쪽으로 이동을 수행
        - 이동하는 수는 상태의 행 번호(0~3)에 의존
          - 0번째 행은 이동되지 않음
          - 1번째 행은 한 자리 왼쪽으로 이동
          - 2번째 행은 두 자리 왼쪽으로 이동
          - 3번째 행은 세 자리 왼쪽으로 이동

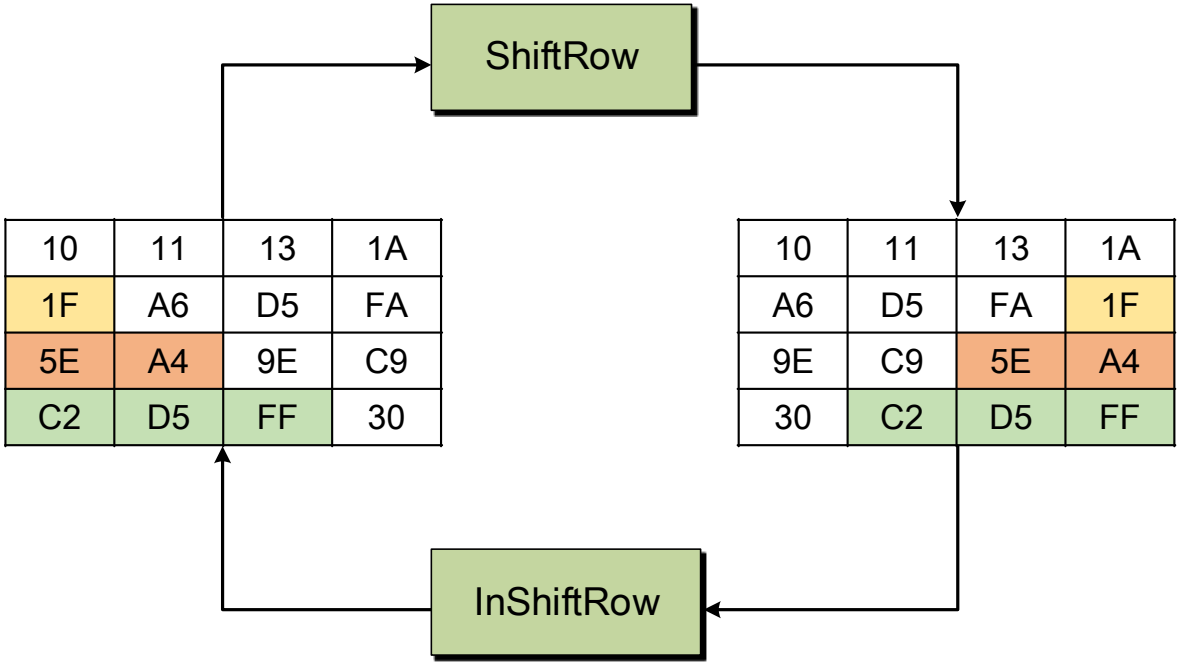


    - InShiftRows
      - 복호화 과정에서 사용
      - 바이트 단위로 오른쪽으로 이동을 수행
        - 이동하는 수는 상태의 행 번호(0~3)에 의존

AES

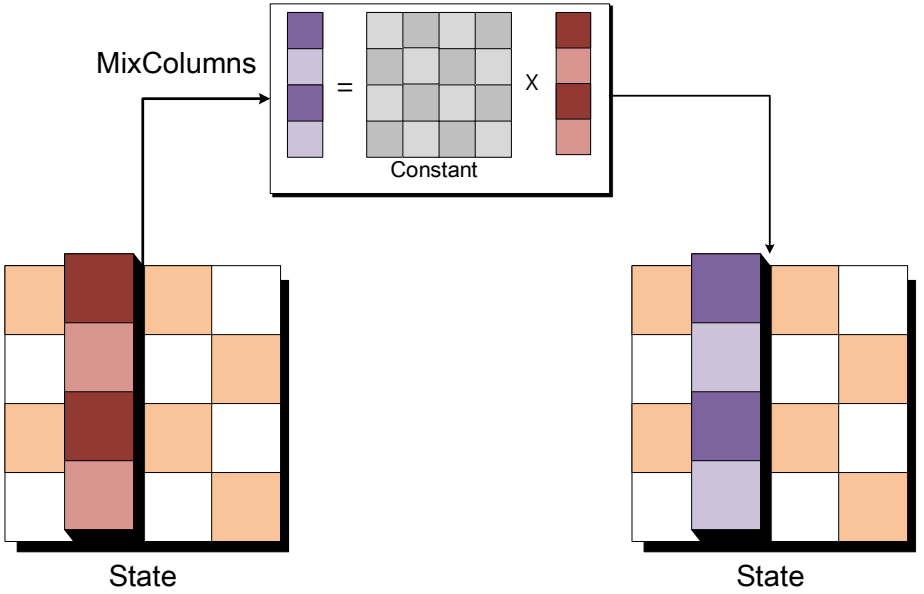
- ShiftRows
  - 예제 7.4

해당 그림은 어떻게 하나의 상태가 ShiftRows 변환을 사용하는지 보여줌



AES

- 라운드(Round)
  - 뒤섞음(Mixing)
    - 한 번에 4개의 바이트들을 가지고 각 바이트의 비트들을 바꾸어서 새로운 4개의 바이트를 생성
    - 각 바이트에 서로 다른 상수 값을 곱함
      - 입력한 4개의 바이트가 동일하더라도 새로운 바이트가 서로 다르다는 것을 보장하기 위해 곱함



AES

- 라운드(Round)
  - 뒤섞음(Mixing)
    - MixColumns
      - 상태 배열의 각 열을 고정된 4X4 행렬과 곱하는 과정

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \times \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

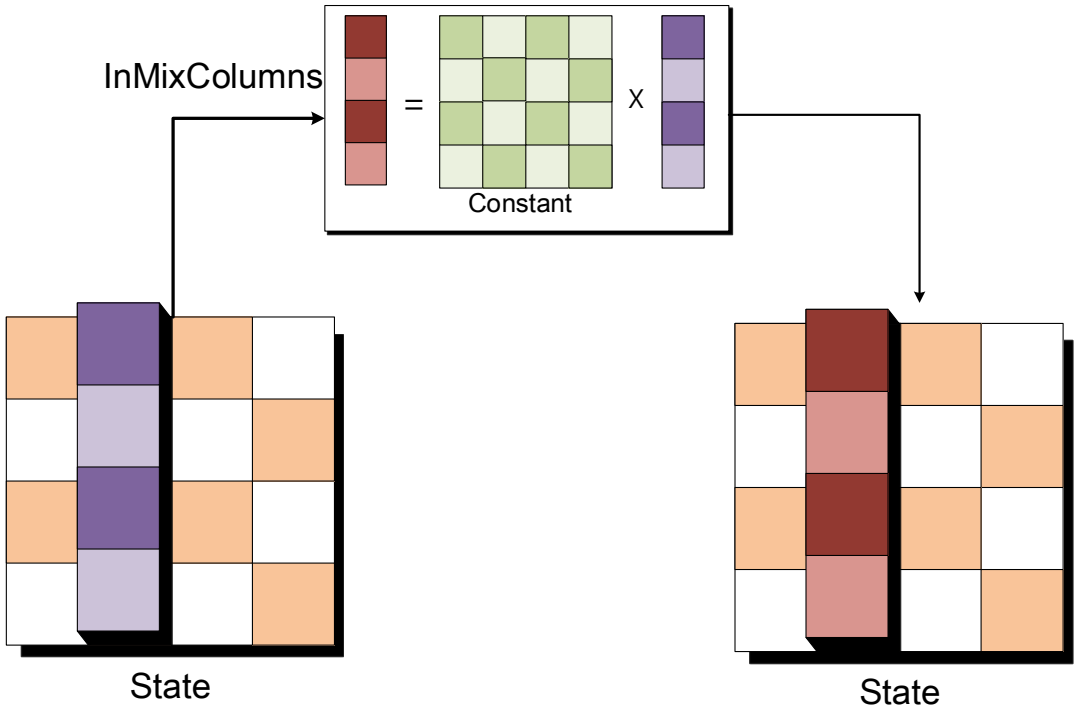
고정된 4×4행렬

$$S'_{0,0} = (02 \cdot S_{0,0}) \oplus (03 \cdot S_{1,0}) \oplus (01 \cdot S_{2,0}) \oplus (03 \cdot S_{3,0})$$
$$S'_{1,0} = (01 \cdot S_{0,0}) \oplus (02 \cdot S_{1,0}) \oplus (03 \cdot S_{2,0}) \oplus (01 \cdot S_{3,0})$$
$$\vdots$$
$$S'_{3,3} = (03 \cdot S_{0,3}) \oplus (01 \cdot S_{1,3}) \oplus (01 \cdot S_{2,3}) \oplus (02 \cdot S_{3,3})$$

AES

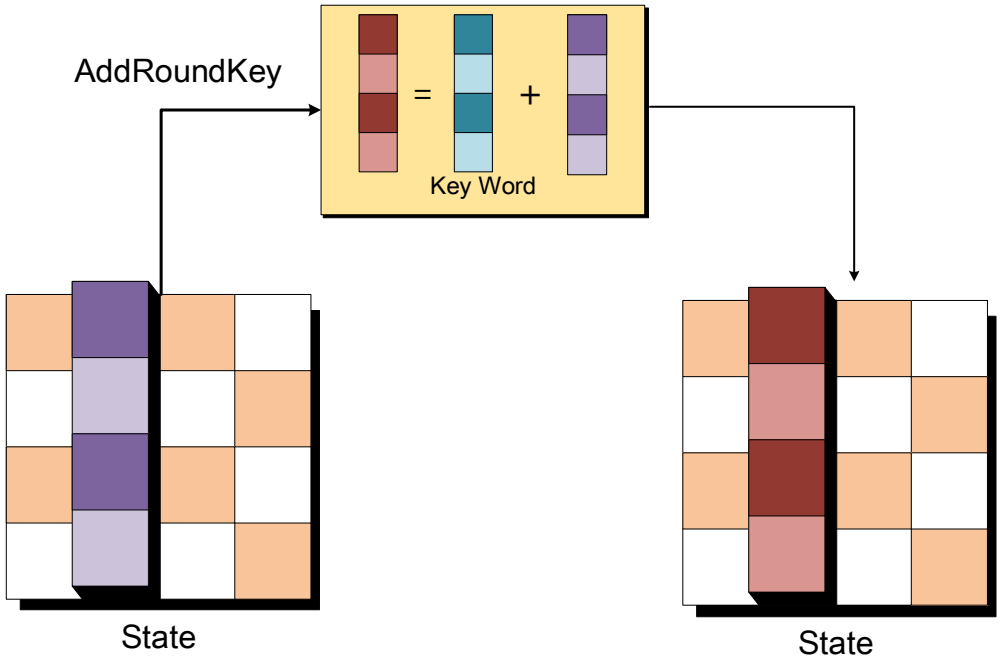
- 라운드(Round)
  - 뒤섞음(Mixing)
    - InMixColumns
      - MixColumns에서 수행된 변환을 역으로 수행
        - 암호화 단계에서 사용된 고정 4X4 행렬과는 다른 행렬을 사용

$$\begin{bmatrix} 0e & 0b & 0\mathrm{d} & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$



AES

- 라운드(Round)
  - 키 덧셈(Key-Adding)
    - AddRoundKey
      - 상태 배열의 각 열에 라운드 키 워드를 더하는 과정(XOR)
      - 자기가 자신의 역변환임
        - 해당 체 위에선 덧셈과 뺄셈이 동일하기 때문



AES

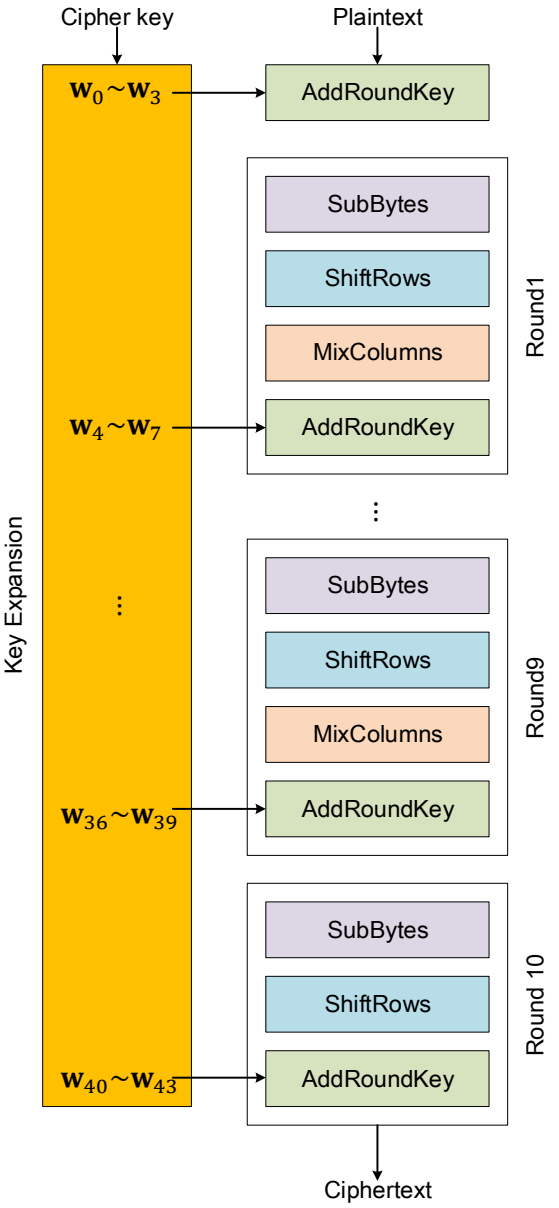
- 구조
  - 요약 (1/2)
    - SubBytes
      - 바이트 치환
        - 고정된 S-박스를 사용해 바이트 단위로 대치
    - ShiftRows
      - 행 이동
        - 상태행렬의 각 행을 왼쪽으로 순환이동
        - 두 번째 행은 1 byte, 세 번째 행은 2 byte, 네 번째 행은 3 byte

AES

- 구조
  - 요약 (2/2)
    - Mixcolumns
      - 열 혼합
        - 상태 행렬의 각 열을 특정 다항식(고정된 행렬과 곱셈)을 이용하여 선형 변환
        - 마지막 라운드에서 해당 단계 생략
    - AddRoundKey
      - 라운드 키 추가
      - 따로 역관계가 정의되지 않음

AES

- 구조
  - 단계
    - (1단계)
      - 평문 블록을 상태 배열로 초기화
    - (2단계)
      - 초기 비밀 키를 여러 라운드 키로 확장
    - (3단계)
      - 라운드 실행
        - SubBytes
        - ShiftRows
        - MixColumns
        - AddRoundKey



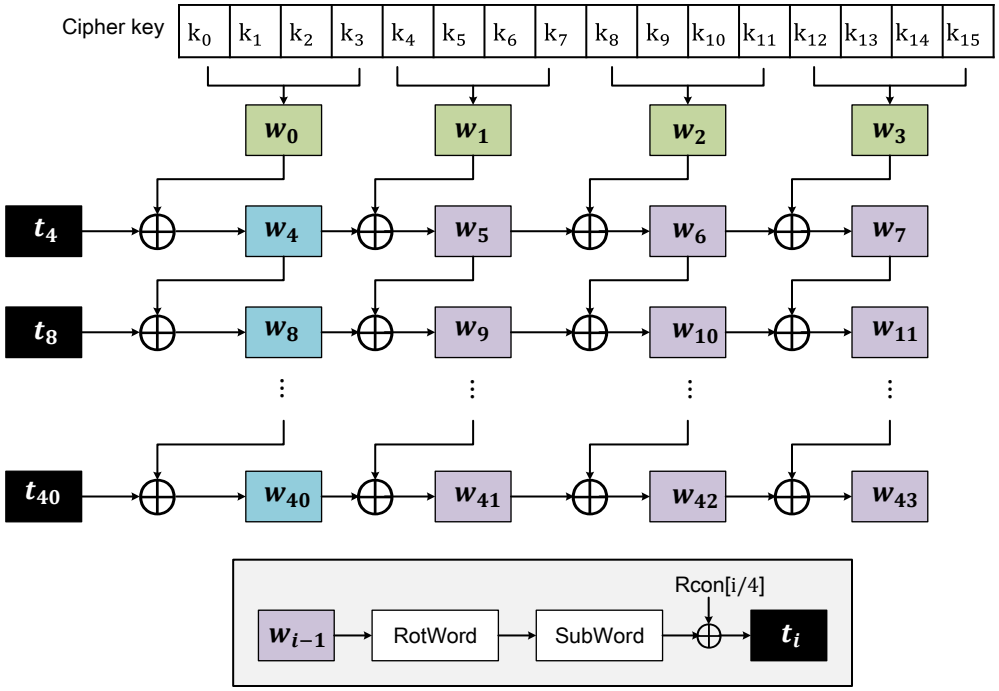
AES

- 구조
  - (2단계) 키 확장 (1/5)
    - n비트 암호 키로 부터 $N_r + 1$개의 n비트 라운드 키를 생성 (N_r은 라운드 수, n은 128, 192, 256 중 하나)
      - 워드가 4개의 바이트들로 이루어진다고 할 때, 각 라운드는 워드 단위로 라운드 키를 생성
        - $4 \times (N_r + 1)$개의 워드를 생성
          - AES-128: 44워드

Round	Words
Pre-Round	$\mathbf{w_0}$ $\mathbf{w_1}$ $\mathbf{w_2}$ $\mathbf{w_3}$
1	$\mathbf{w_4}$ $\mathbf{w_5}$ $\mathbf{w_6}$ $\mathbf{w_7}$
2	$\mathbf{w_8}$ $\mathbf{w_9}$ $\mathbf{w_{10}}$ $\mathbf{w_{11}}$
…	…
N_r	$\mathbf{w_{4N_r}}$ $\mathbf{w_{4N_r+1}}$ $\mathbf{w_{4N_r+2}}$ $\mathbf{w_{4N_r+3}}$

AES

- 구조
  - (2단계) 키 확장 (2/5)
    - e.g., AES-128 키 확장 과정
      - $i \; mod \; 4 \neq 0$일 경우
        - $\boldsymbol{w_i = w_{i-1} \oplus w_{i-4}}$
      - $i \; mod \; 4 = 0$일 경우
        - $\boldsymbol{w_i} = t \oplus \boldsymbol{w_{i-4}}$
        - $t = \mathrm{SubWord}(\mathrm{RotWord}(\mathrm{w_{i-1}})) \oplus \mathrm{RCon_{i/4}}$



AES

- 구조
  - (2단계) 키 확장 (3/5)
    - e.g., AES-128 키 확장 과정
      - SubWord
        - 4바이트 워드의 각 바이트에 S-박스(SubBytes 변환에 사용된 것)를 적용하여 바이트 치환을 수행
      - RotWord
        - 4바이트 워드의 바이트들을 순환 이동하여, 왼쪽으로 한 바이트씩 이동 수행
          - e.g., $[k_0,\ k_1,\ k_2,\ k_3] \rightarrow [k_1,\ k_2,\ k_3, k_0]$

AES

- 구조
  - (2단계) 키 확장 (4/5)
    - e.g., AES-128 키 확장 과정
      - RCon
        - $\mathrm{RC_i}$라고 불리는 라운드 상수는 4바이트 값이며, 표 또는 체 $GF(2^8)$를 이용하여 상수들의 최상위 바이트들을 계산하여 얻을 수 있음
        - $\mathrm{RC_i} \rightarrow x^{i-1}$로 표현 ($prime$는 기약 다항식 $x^8 + x^4 + x^3 + 1$)
          - e.g., $\mathrm{RC_1} \rightarrow x^{1-1} = x^0 \, mod \, prime = 1 \rightarrow 00000001 \rightarrow 01_{16}$
            $\mathrm{RC_9} \rightarrow x^8 \, mod \, prime = x^4 + x^3 + x + 1 \rightarrow 00011011 \rightarrow 1B_{16}$

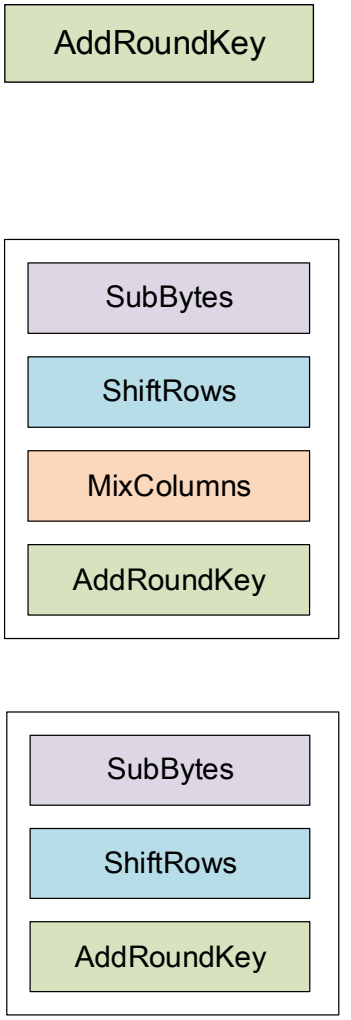
Round	RCon	Round	RCon
1	$(\mathbf{01}\ 00\ 00\ 00)_{16}$	6	$(\mathbf{20}\ 00\ 00\ 00)_{16}$
2	$(\mathbf{02}\ 00\ 00\ 00)_{16}$	7	$(\mathbf{40}\ 00\ 00\ 00)_{16}$
3	$(\mathbf{04}\ 00\ 00\ 00)_{16}$	8	$(\mathbf{80}\ 00\ 00\ 00)_{16}$
4	$(\mathbf{08}\ 00\ 00\ 00)_{16}$	9	$(\mathbf{1B}\ 00\ 00\ 00)_{16}$
5	$(\mathbf{10}\ 00\ 00\ 00)_{16}$	10	$(\mathbf{36}\ 00\ 00\ 00)_{16}$

AES

- 구조
  - (2단계) 키 확장 (5/5)
    - 키 확장 과정 특징
      - 공격자가 암호 키 일부나 라운드 키의 일부 값을 얻는다고 해도 모든 라운드 키를 알기 위해서는 남은 암호 키를 복구해야함
        - SubWord의 비선형성 때문
      - DES와 달리 심각한 취약 키를 가지지 않음
        - S-박스의 비선형성이나 ShiftRows와 MixColumns 단계에서 데이터의 비트와 바이트 단위로 섞기 때문
        - DES는 Feistel 구조를 사용해 각 라운드가 독립적이지 않아 특정 패턴의 키에 대해 취약할 수 있음

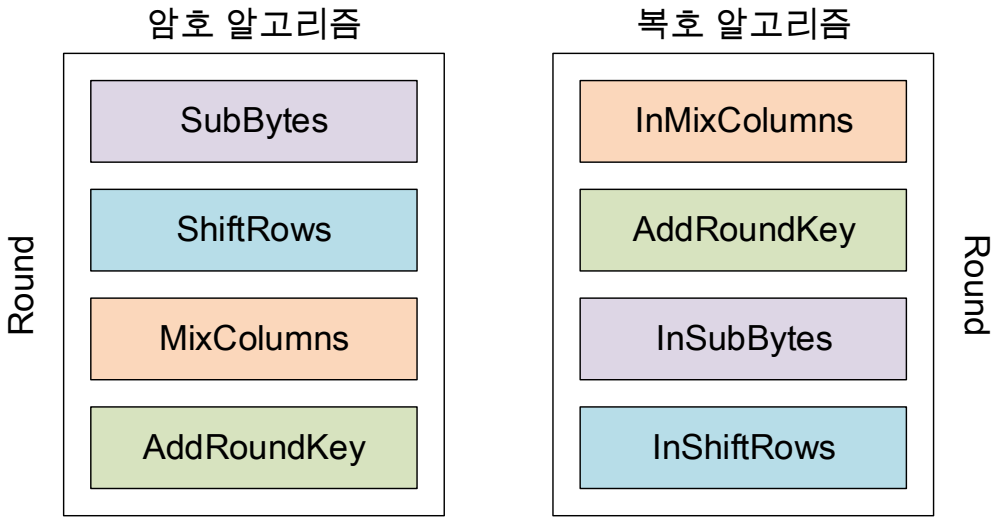
AES

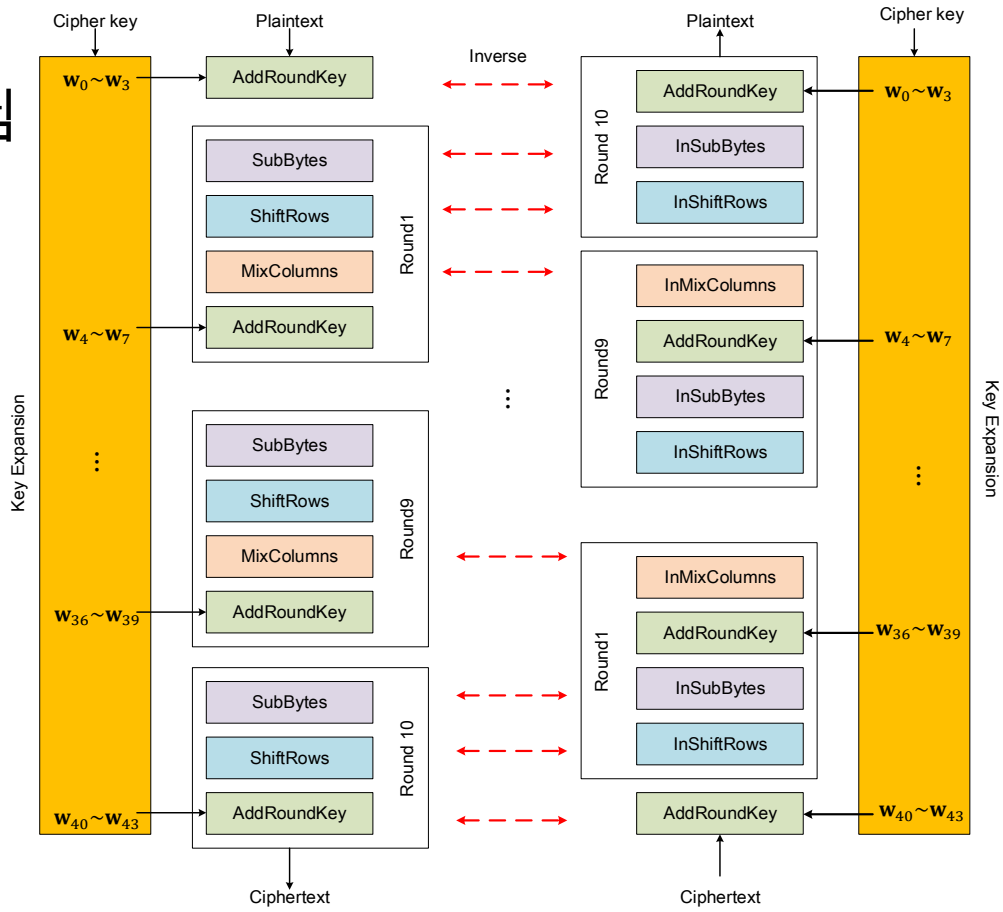
- 구조
  - (3단계) 라운드 수행
    - 라운드 구조
      - 초기 라운드 (1회)
        - AddRoundKey
      - 일반 라운드 ($N_r - 1$회)
        - SubBytes
          ShiftRows
          MixColumns
          AddRoundKey
      - 최종 라운드 (1회)
        - SubBytes
          ShiftRows
          AddRoundKey



AES

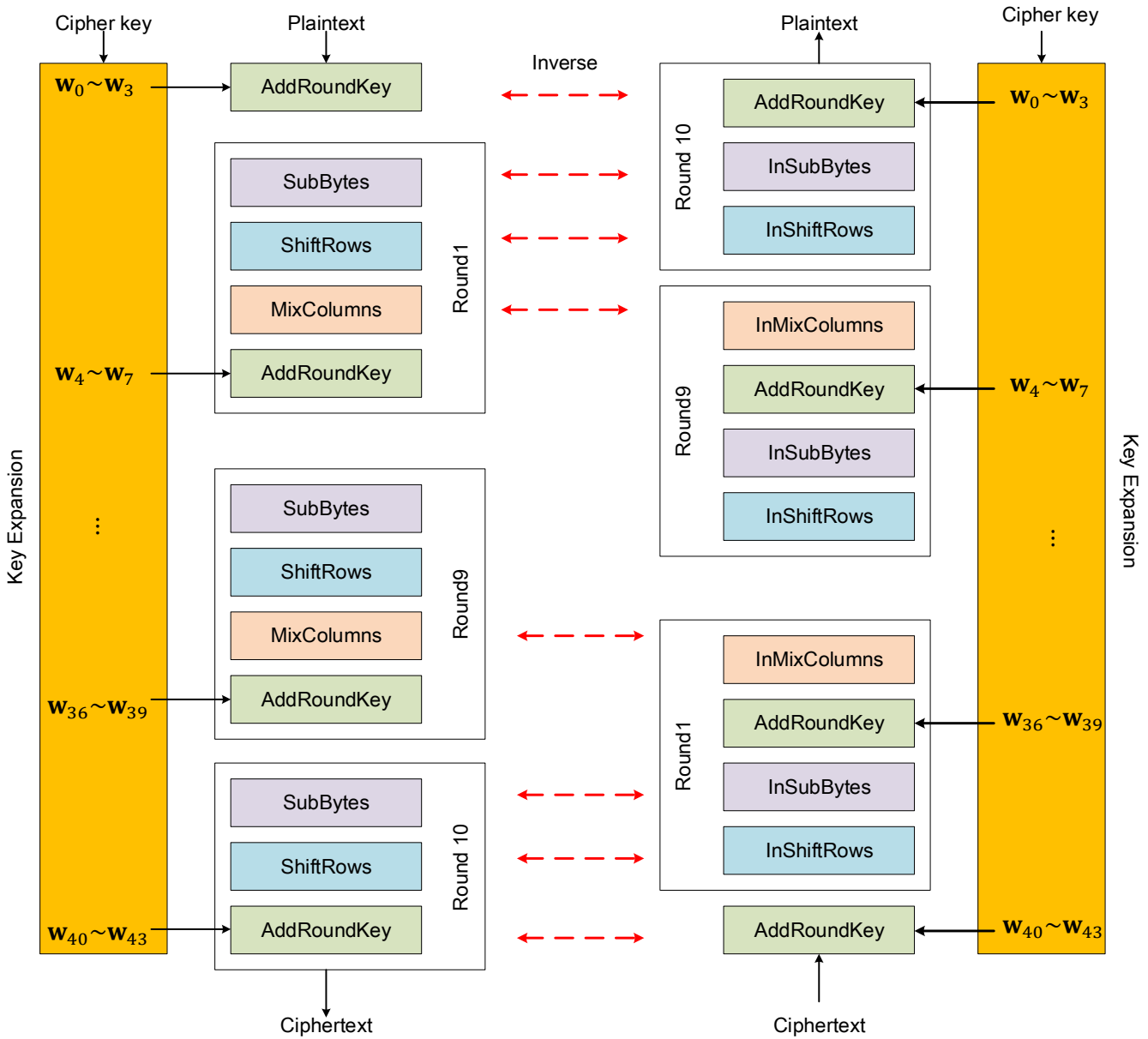
- 구조
  - 설계 특징
    - 복호 알고리즘은 암호 알고리즘의 역함수
      - 대치와 치환의 순서가 바뀜
      - 뒤섞임과 키 덧셈 순서가 바뀜





AES

- 구조

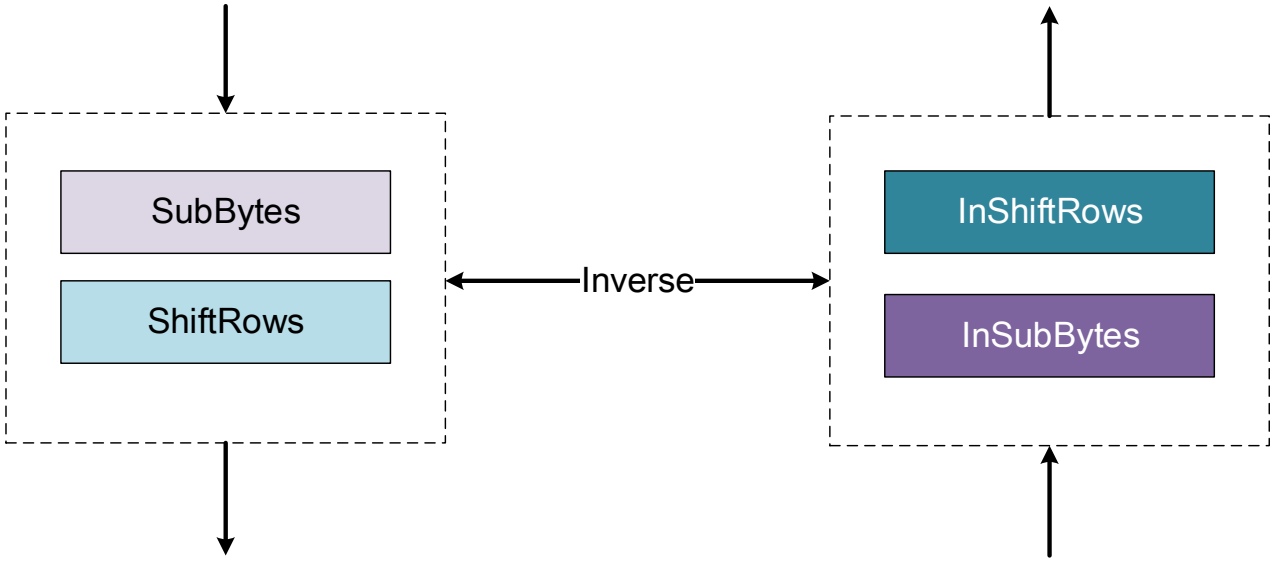

Cipher key
Plaintext
Inverse
Plaintext
Cipher key
$w_0 \sim w_3$
AddRoundKey
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round1
Round 10
AddRoundKey
InSubBytes
InShiftRows
$w_4 \sim w_7$
Round9
InMixColumns
AddRoundKey
InSubBytes
InShiftRows
Key Expansion
$w_{36} \sim w_{39}$
Round1
InMixColumns
AddRoundKey
InSubBytes
InShiftRows
$w_{40} \sim w_{43}$
Round 10
AddRoundKey
ShiftRows
SubBytes
AddRoundKey
Ciphertext
Ciphertext

AES

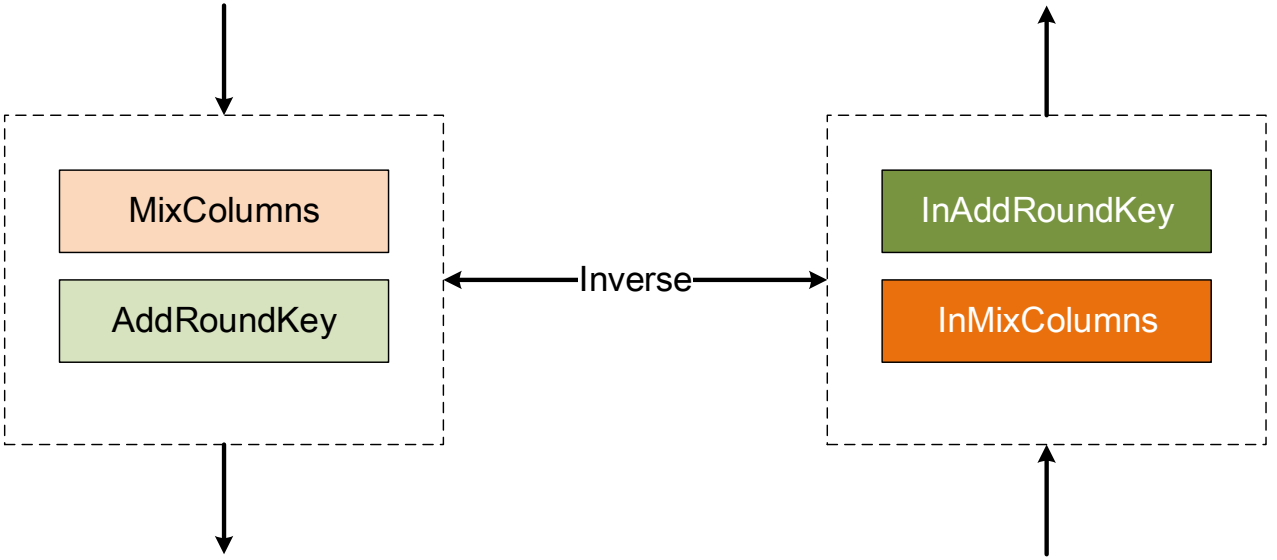
- 구조
  - 다른 설계 버전 (1/3)
    - 각 변환 성질을 가진 함수들의 순서를 바꾸지 않은 버전
      - 이 설계는 하나의 변환이 아니라 변환들의 쌍에 대해서 성립하여야 함
      - 암호와 역암호에서 두 변환의 조합은 서로 역함수 관계에 있어야 함

AES

- 구조
  - 다른 설계 버전 (2/3)
    - SubBytes와 ShiftRow
      - SubBytes에서 바이트의 순서를 변경하는 것이 아니라 바이트 값을 변경
      - ShiftRows에서 바이트 값을 변경하지 않고 순서만 변경

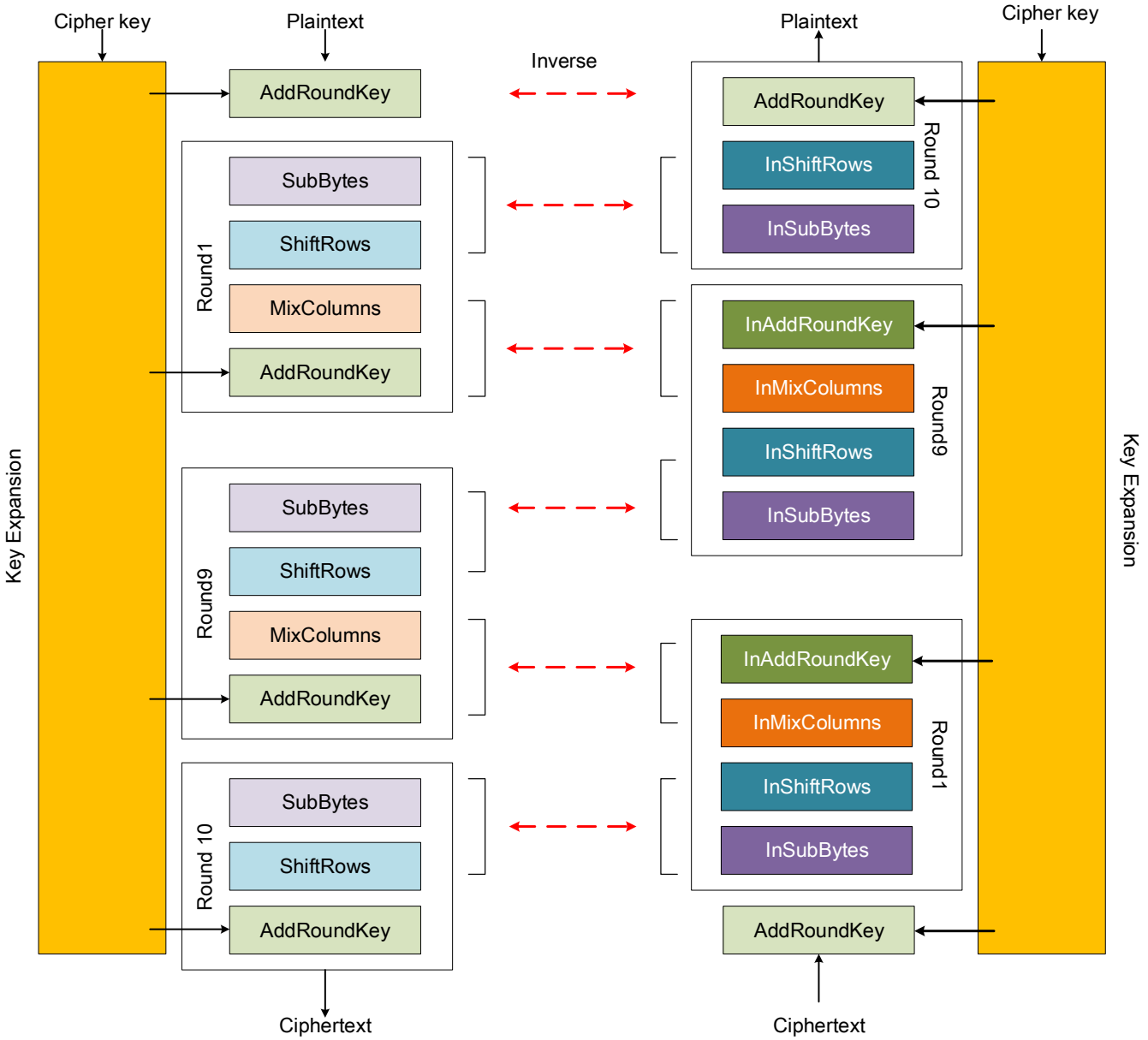


- 구조
  - 다른 설계 버전 (3/3)
    - MixColumns과 AddRoundKey
      - MixColumns에서 사용된 상수 행렬의 역 행렬(X^{-1})을 키 행렬과 곱하여 생성된 새로운 변환을 InAddRoundKey라고 부름



AES

- 구조
  - 다른 설계 버전



AES

- 분석
  - 안전성
    - DES의 경우, 올바른 키를 찾는데 2^{56}번의 테스트가 필요하며 t초가 걸리지만 AES의 경우, 2^{128}번 이상의 테스트가 필요하며 $2^{72} \times t$ 초가 걸림
    - AES는 소프트웨어, 하드웨어, 펌웨어로 구현될 수 있음
      - 테이블 참조 또는 잘 정의된 대수구조를 사용하여야 구현가능
      - 변환은 바이트, 워드 기반으로 설계 가능
    - DES와 달리 차분 공격과 선형 공격이 알려지지 않음
      - 설계 당시, 이 공격에 대한 안전성을 고려하여 설계함

부록

	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

첫 번째 S박스

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

네 번째 S박스

부록

	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	10	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	09	03	05	06	11

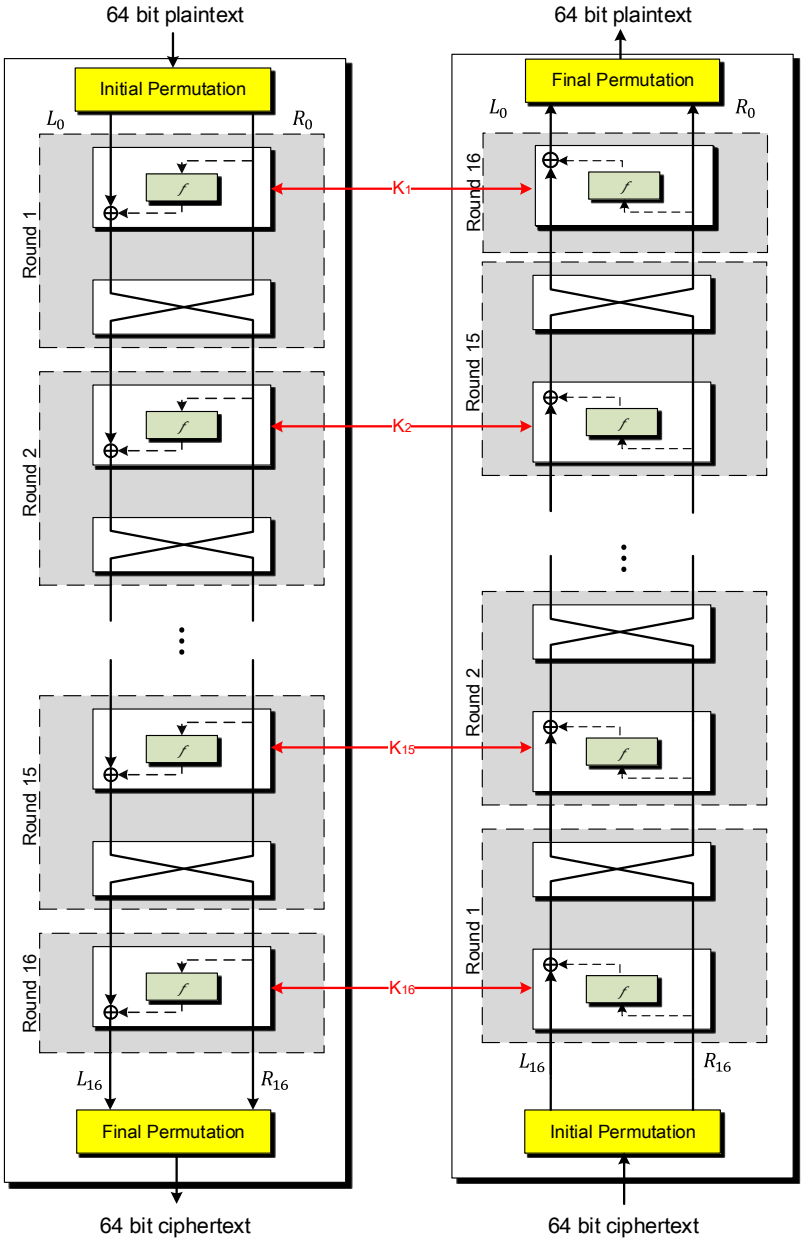
여덟 번째 S박스

부록

- SubBytes

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
01	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
02	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
03	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
04	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
05	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
06	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
07	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
08	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
09	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

부록


64 bit plaintext
Initial Permutation
L_0
R_0
Round 1
Round 2
Round 15
Round 16
K_1
K_2
K_{15}
K_{16}
L_{16}
R_{16}
Final Permutation
64 bit ciphertext
64 bit plaintext
Final Permutation
L_0
R_0
Round 16
Round 15
Round 2
Round 1
L_{16}
R_{16}
Initial Permutation
64 bit ciphertext

Thanks!

김 혜 정(hyejeong@pel.sejong.ac.kr)