

제 4 장 대칭 암호



박 종 혁 교수

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

Chapter 4 대칭 암호

1절 문자 암호에서 비트열 암호로

2절 일회용 패드-절대 해독 불가능한 암호

3절 DES란?

4절 트리플 DES

5절 AES 선정 과정

6절 Rijndael

제1절 문자 암호에서 비트열 암호로

1.1 부호화

1.2 XOR

1.1 부호화

- 암호화에 컴퓨터 사용이 필수
- 암호화 프로그램도 평문을 비트열로 변경하고 비트열로 된 암호문을 출력
- 부호화(encoding)
 - 문자열을 비트열로 바꾸는 것

ASCII

- 문자열 midnight 을 다음과 같은 비트열로 부호화

m → 01101101

i → 01101001

d → 01100100

n → 01101110

i → 01101001

g → 01100111

h → 01101000

t → 01110100

1.2 XOR

- XOR은 '익스클루시브 오아(exclusive or)', 또는 짧게 '엑스오아'라고 읽는다.

- 우리 말로는 배타적 논리합

$0 \text{ XOR } 0 = 0$ (0과 0의 XOR은 0이 된다)

$0 \text{ XOR } 1 = 1$ (0과 1의 XOR은 1이 된다)

$1 \text{ XOR } 0 = 1$ (1과 0의 XOR은 1이 된다)

$1 \text{ XOR } 1 = 0$ (1과 1의 XOR은 0이 된다)

한 비트의 XOR

- XOR은 $\oplus$ 이라는 기호를 써서 표현

$a \oplus b$	설명
$0 \oplus 0 = 0$	0과 0의 XOR은 0이 된다
$0 \oplus 1 = 1$	0과 1의 XOR은 1이 된다
$1 \oplus 0 = 1$	1과 0의 XOR은 1이 된다
$1 \oplus 1 = 0$	1과 1의 XOR은 0이 된다

- 같은 숫자끼리의 XOR은 반드시 0이 된다

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

비트열 XOR

$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ \cdots\ A \\ \oplus\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ \cdots\ B \\ \hline 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ \cdots\ A\ \oplus\ B \end{array}$$

비트열 XOR

$$\begin{array}{r} 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ \dots\ A \oplus B \\ \oplus\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ \dots\ B \end{array}$$

$$0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ \dots\ A \oplus B \oplus B = A$$

(A로 돌아간다)

암호화/복호화의 순서와 매우 비슷

- 평문 A 를 키 B 로 암호화하고, 암호문 $A \oplus B$ 를 얻는다.
- 암호문 $A \oplus B$ 를, 키 B 로 복호화해서 평문 A 를 얻는다.

XOR은 그림을 마스크한다



XOR은 그림을 마스크한다

제2절 일회용 패드-절대 해독 불가능한 암호

2.1 일회용 패드란?

2.2 일회용 패드의 암호화

2.3 일회용 패드의 복호화

2.4 일회용 패드는 해독할 수 없다

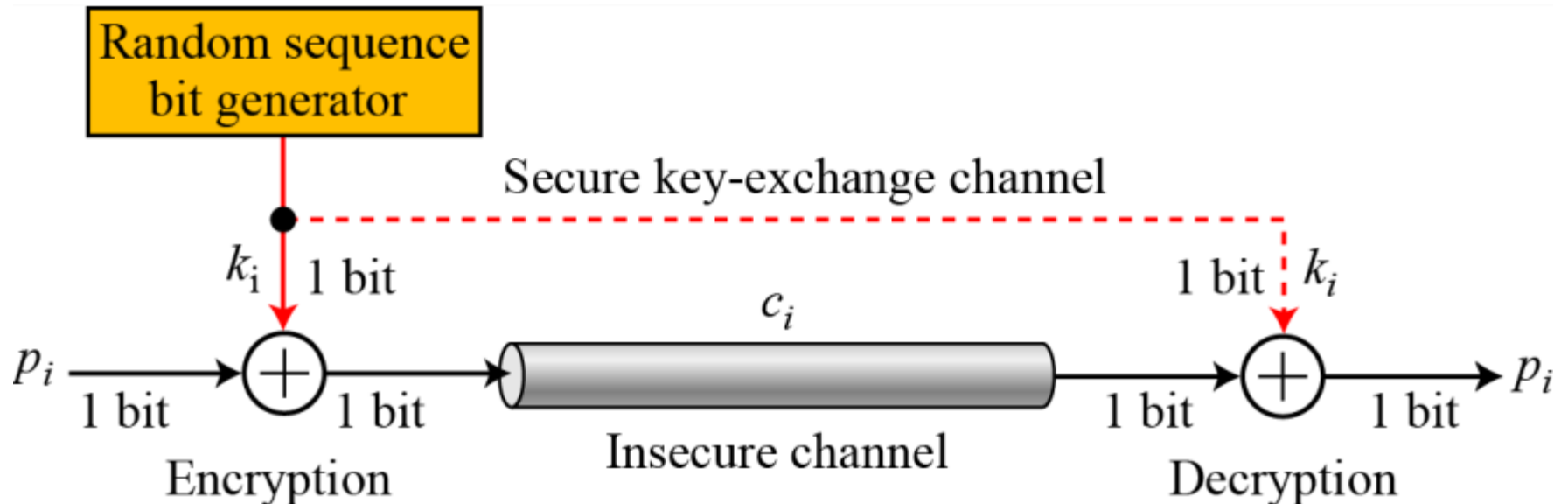
2.5 일회용 패드는 왜 사용되지 않을까?

2.1 일회용 패드란?

- 1회용 패드(one-time pad)
 - 전사공격에서 키공간을 모두 탐색하더라도 해독할 수 없는 암호

2.2 일회용 패드의 암호화

- 평문과 랜덤한 비트열과의 XOR만을 취하는 단순한 암호



일회용 패드 암호화 예

- 평문: midnight
– ASCII로 부호화

문자	m	i	d	n	i	g	h	t
ASCII 코드	01101101	01101001	01100100	01101110	01101001	01100111	01101000	01110100
단어	m i d n i g h t							

- 키: 랜덤 비트열

키	01101011	11111010	01001000	11011000	01100101	11010101	10101111	00011100
---	----------	----------	----------	----------	----------	----------	----------	----------

일회용 패드 암호화 예

	01101101	01101001	01100100	01101110	01101001	01100111	01101000	01110100	midnight
$\oplus$	01101011	11111010	01001000	11011000	01100101	11010101	10101111	00011100	키
	00000110	10010011	00101100	10110110	00001100	10110010	11000111	01101000	암호문

2.3 일회용 패드의 복호화

- 암호문과 키의 XOR을 계산하면 평문

	00000110	10010011	00101100	10110110	00001100	10110010	11000111	01101000	암호문
$\oplus$	01101011	11111010	01001000	11011000	01100101	11010101	10101111	00011100	키
	01101101	01101001	01100100	01101110	01101001	01100111	01101000	01110100	midnight

2.4 일회용 패드는 해독할 수 없다

- 현실적인 시간 내에 해독이 곤란하다는 의미만은 아니다.
- 키 공간 전체를 순식간에 계산할 수 있는 무한대의 계산력을 갖는 컴퓨터로도 일회용 패드는 해독할 수 없다.
- 문자열이 복호화 되었다 하더라도, 그것이 바른 평문인지 아닌지 판정할 수 없다.

전사 공격

- 암호문을 복호화해보면 도중에 모든 64비트 패턴이 등장한다
- 그 중에 나타날 수 있는 문자열
 - 규칙적인 문자열
aaaaaaaa, abcdefgh, zzzzzzzz 등
 - 의미 있는 영어 단어
midnight, onenight, mistress 등
 - 무의미한 문자열
%Ta_AjvX, HY(&JY!z, \$@~*W^^), Er#f6)(%
- 따라서 어느 것이 바른 평문인지 알 수 없다
 - 즉 어떤 키를 사용하면 바르게 복호화할 수 있는지 알 수 없다

전사 공격

- 일회용 패드에서는 키들을 적용하여 얻어진 것이 바른 평문인지 아닌지를 판정하는 것이 불가능하다.
- 그러므로 일회용 패드를 해독할 수 없다.

2.5 일회용 패드는 왜 사용되지 않을까?

- 키 배송, 키 보존, 키 재사용, 키 동기화, 키 생성 이슈가 발생
- 일회용 패스의 사용은 키 생성과 배송에 막대한 비용과 노력이 필요
 - 비용과 상관없이 기밀성 최우선 고려하는 경우만 사용
예) 강대국끼리의 핫라인
- 실제 스트림 암호에 활용

Question (1# Home Work)

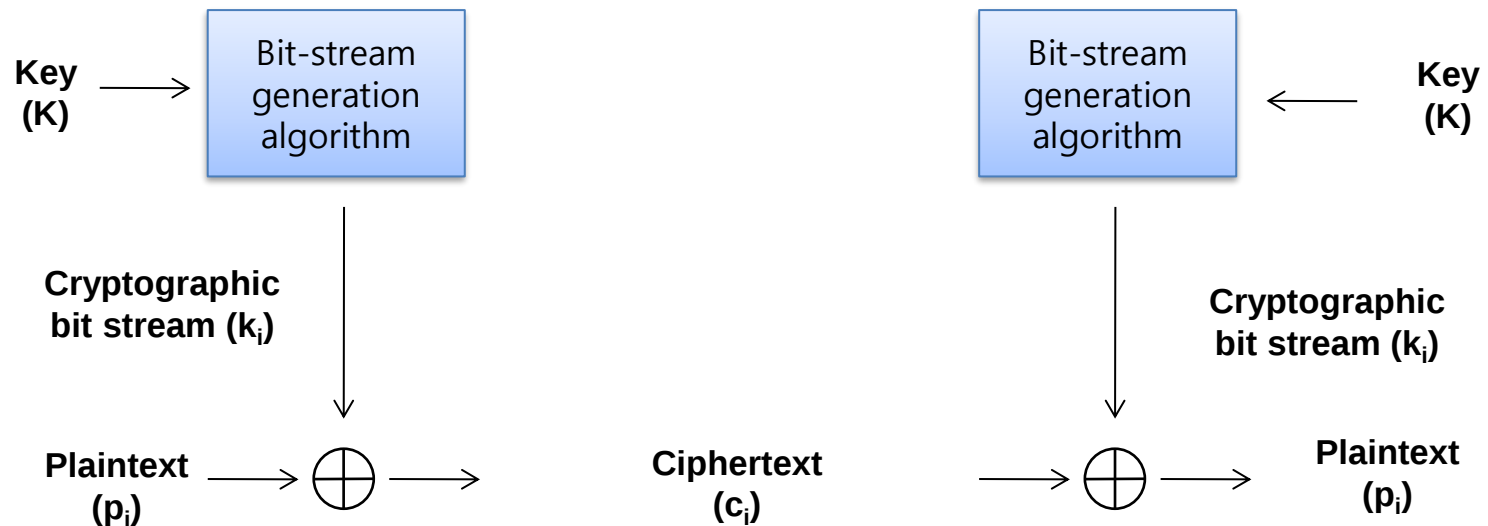
- 다음 경우 각각에 대하여 one-time pad 암호의 암호문의 패턴은 무엇인가?
 - a. 평문이 n 개의 0으로 구성되는 경우
 - b. 평문이 n 개의 1으로 구성되는 경우
 - c. 평문이 0과 1로 교차되어 구성되는 경우
 - d. 평문이 랜덤 스트림 비트인 경우

스트림 암호와 블록암호

블록 암호 기법의 원리

- 스트림 암호

- 한 번에 1비트 혹은 1바이트의 디지털 데이터 스트림을 암호화 하는 방식



블록 암호 기법의 원리

- 블록 암호

- 평문 블록 전체를 가지고 같은 크기의 암호문 블록을 생성
- 모드를 이용하여 스트림 암호 기법과 동일한 효과

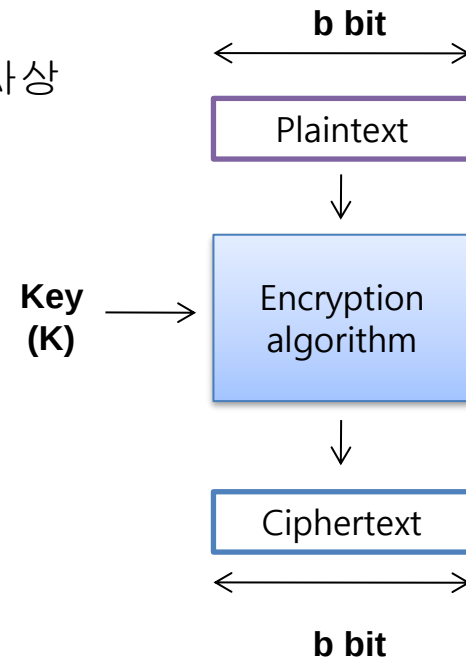
- 두 개 이상의 기본 암호를 연속적으로 수행(치환, 순열 번갈아 수행)

- 치환 (substitution)

- .평문의 각 원소 또는 원소의 그룹을 다른 원소에 사상
- .혼돈의 원리를 이용

- 순열(permutation)

- .평문 원소의 순서는 순열의 순서대로 재배치
- .확산의 원리를 이용



블록 암호 기법의 원리

- Feistel 암호 방식 (혼돈과 확산: Confusion & Diffusion)

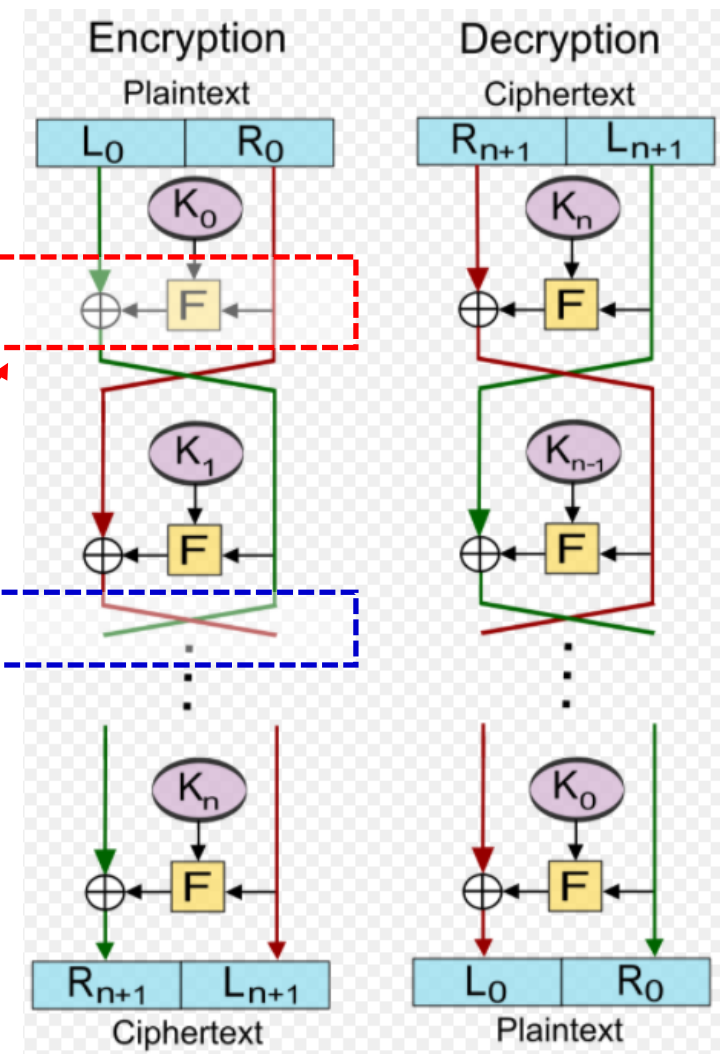
- Claude Shannon 소개(SHAN49): “매우 이상적인 암호는 암호문에 대한 모든 통계적 정보가 사용된 키와 독립적이어야 한다.”
- 통계적 분석에 기초한 암호 해독 방지

- **혼돈** : 키를 발견하기 어렵게 하기 위해 **암호문**에 대한 통계 값과 **암호 키 값 사이**에 관계를 가능한 복잡하게 하는 것

- **확산** : **평문**의 통계적 구조가 **암호문**의 광범위한 통계값에 **분산**되도록 하여
 - 키를 추론하기 어렵게 하기 위해 평문과 암호문 사이에 통계적인 관계를 가능한 복잡하게 만드는 것

3.4 Feistel 암호 구조

- 처리구조
 - 길이 $2w$ 비트인 평문 블록 (L_0, R_0) 분할 처리
 - K 로 부터 유도된 n 개의 키(K_i) 사용
 - n 회의 동일한 반복 구조 실행
- 그림: 고전 Feistel 구조
- 하나의 반복 구조
 - 오른 쪽 반 R_0 에 반복 함수 F 적용
 - ❖ 반복 서브키 K_1 적용 ($K \neq K_i$)
 - 왼쪽 반 L_0 와 XOR
 - (치환: Substitution 작용)
 - 좌우 양쪽 결과를 교환
 - (순열: Permutation 작용)



Feistel 네트워크의 매개 변수와 설계 특성

- 블록 크기
 - 64비트가 일반적, 현재 가변적
- 키 크기
 - 64비트 또는 128비트
- 반복 수
 - Feistel 암호 방식은 다중 반복 과정은 보안성을 증가
 - 일반화는 16회
- 서브키 생성 알고리즘
 - 알고리즘이 복잡할수록 암호해독이 더욱 더 어려움
- 반복 함수
 - 함수가 복잡할 수록 일반적으로 암호해독이 더욱 더 어려움
- 빠른 소프트웨어 암호/복호화
 - 프로그램의 실행 속도가 관심사
- 분석의 용이성

Feistel 네트워크의 특성

- 원하는 만큼 라운드수를 늘릴 수 있음
- 라운드 함수 F 에 어떤 함수를 사용해도 복호화가 가능
- 암호화와 복호화를 완전히 동일한 구조로 실현할 수 있음

제3절 DES란?

3.1 DES란?

3.2 DES 암호화/복호화

3.3 DES의 구조

3.4 Feistel 구조

3.5 차분 해독법과 선형 해독법

3.1 DES 란?

- DES(Data Encryption Standard)는 1977년에 미국의 연방 정보 처리 표준 규격(FIPS)으로 채택된 대칭 암호
- 전사 공격으로 해독 할수 있는 수준

3.1 DES 란?

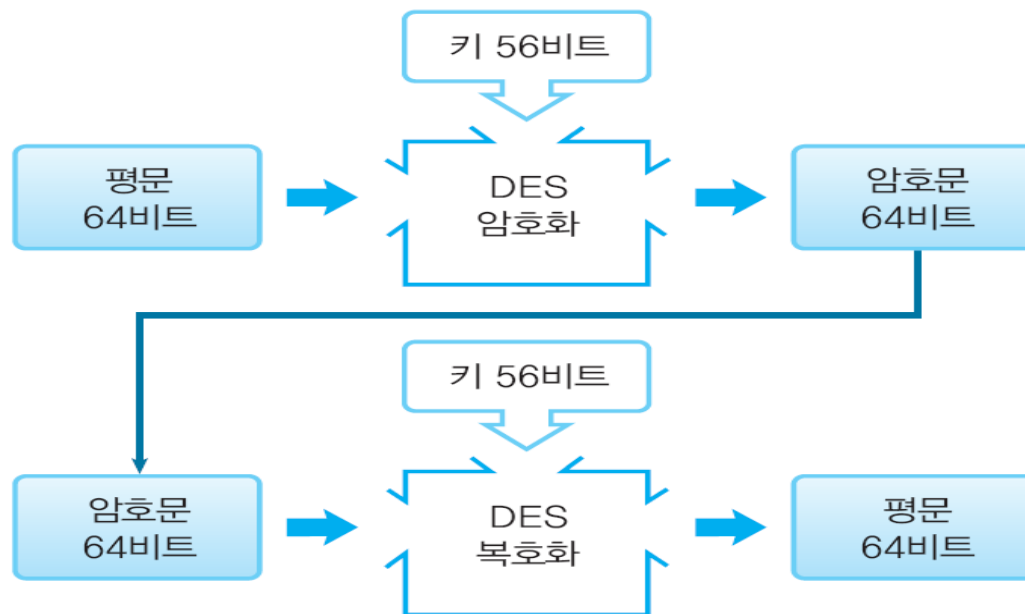
- DES는 64비트의 키를 적용하여 64비트의 평문을 64비트의 암호문으로 암호화 시키는 대칭형 블록 암호이다.
- DES 알고리즘에서는 사용하는 함수
 - 치환(substitution)
 - 순열(permutation)
- 대체와 치환은 1949년도에 Claude Shannon이 제시한 혼돈(confusion)과 확산(diffusion)이라는 두 가지 개념에 기반을 두고 있다.

DES 콘테스트(DES Challenge)

- 1997년의 DES Challenge I
 - 96일
- 1998년의 DES Challenge II -1
 - 41일
- 1998년의 DES Challenge II -2
 - 56시간
- 1999년의 DES Challenge III
 - 22시간 15분

3.2 DES의 암호화 · 복호화

- 64비트 평문을 64비트 암호문으로 암호화하는 대칭 암호 알고리즘
- 키의 비트 길이는 56비트
- 64비트 평문(비트열)을 하나의 단위로 모아서 암호화



DES의 암호화 · 복호화

3.3 DES의 구조

• 단순 DES

- 교육용 알고리즘으로 손으로 예제를 풀 수 있음
- 단순 DES는 S-DES라고 표현
 - 8비트 평문 블록 (1 0 1 1 1 1 0 1)
 - 10비트 키를 입력

암호알고리즘

$$IP^{-1} \circ f_{K_2} \circ SW \circ f_{K_1} \circ IP$$

다른 표현

$$\text{ciphertext} = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(\text{plaintext}))))))$$

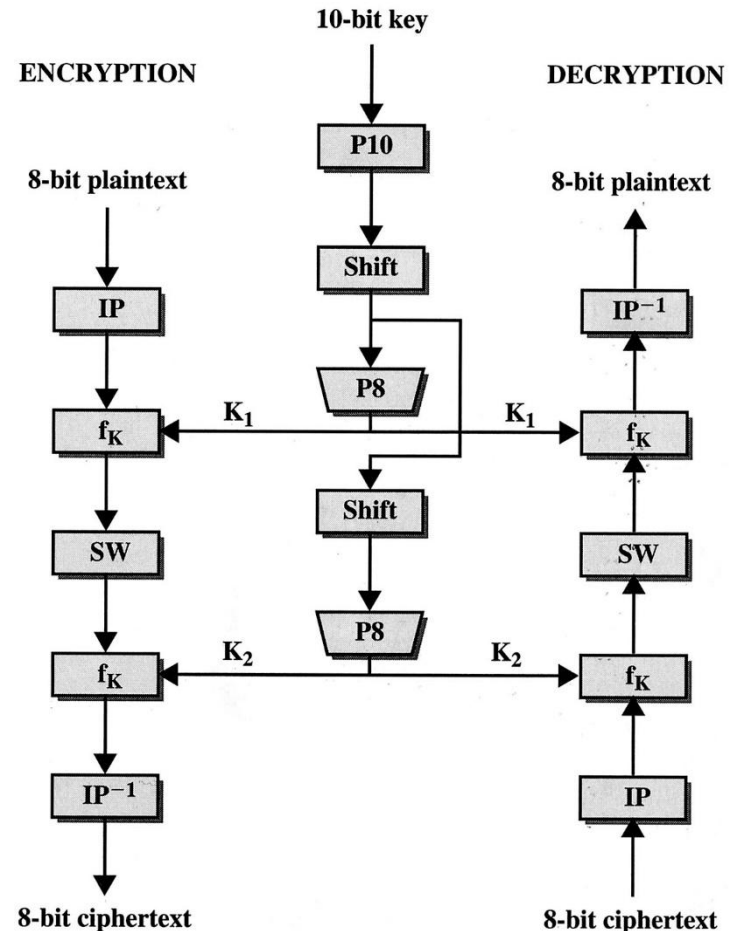
K에 대한 표현

$$K_1 = P8(\text{Shift}(P10(\text{Key})))$$

$$K_2 = P8(\text{Shift}(\text{Shift}(P10(\text{Key}))))$$

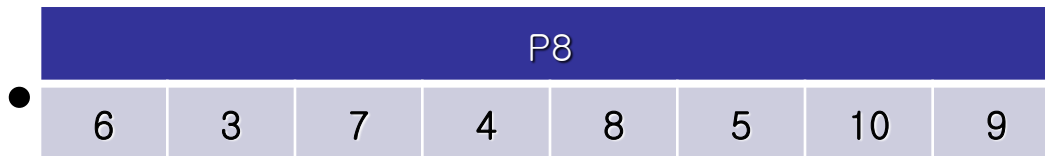
복호화알고리즘

$$\text{Plaintext} = IP^{-1}(f_{K_1}(SW(f_{K_2}(IP(\text{ciphertext}))))))$$



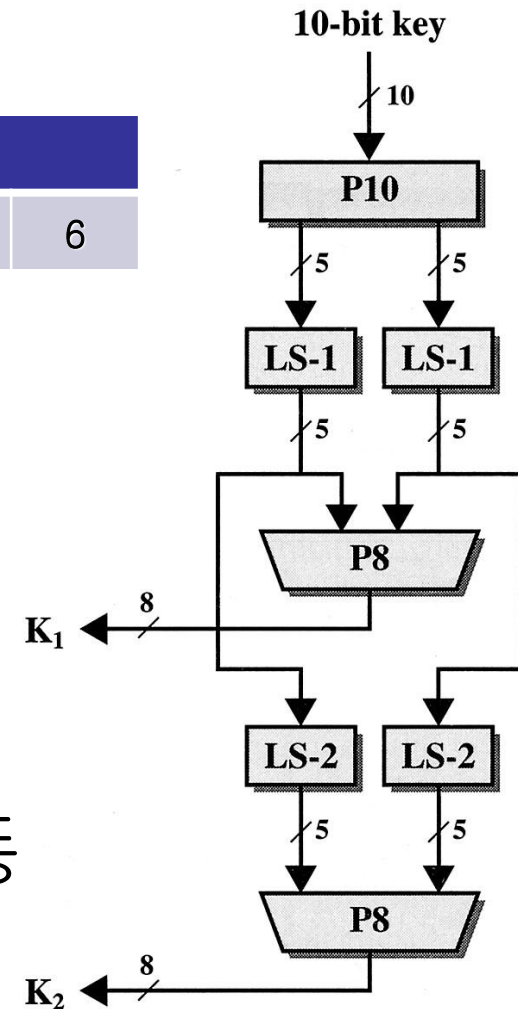
3.3 DES의 구조

- 키의 생성 P10

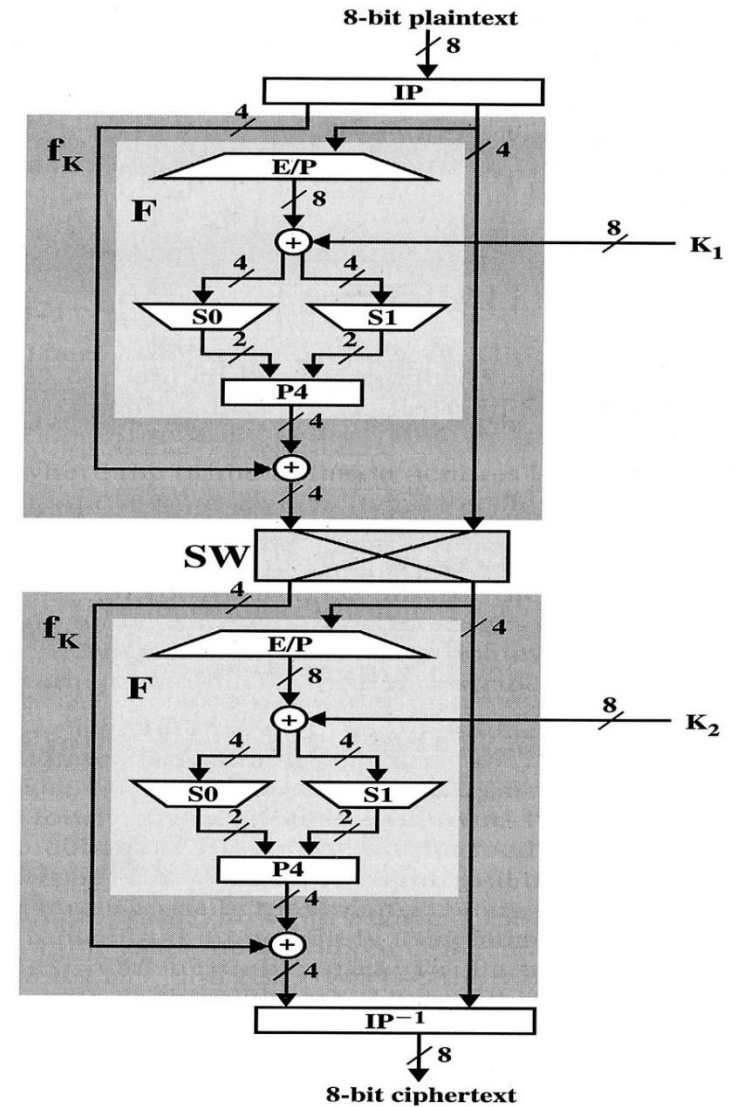
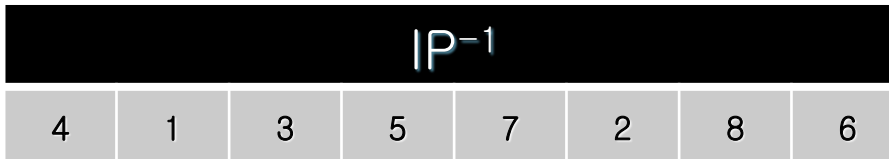
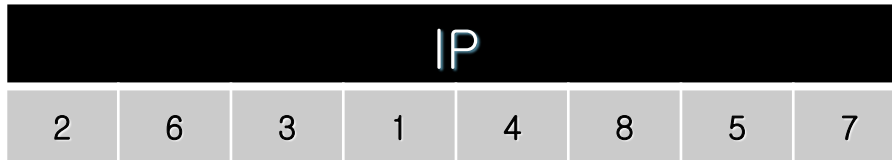


- LS는 비트를 좌로 순환 이동

- LS-1은 한 비트 좌로 이동
- LS-2은 두 비트 좌로 이동



3.3 DES의 구조



3.3 DES의 구조

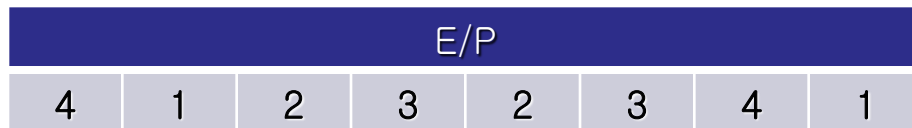
- 함수 F

$$f_k(L, R) = (L \oplus F(R, SK), R)$$

SK: 서브키

$\oplus$: 비트별 배타적 논리합

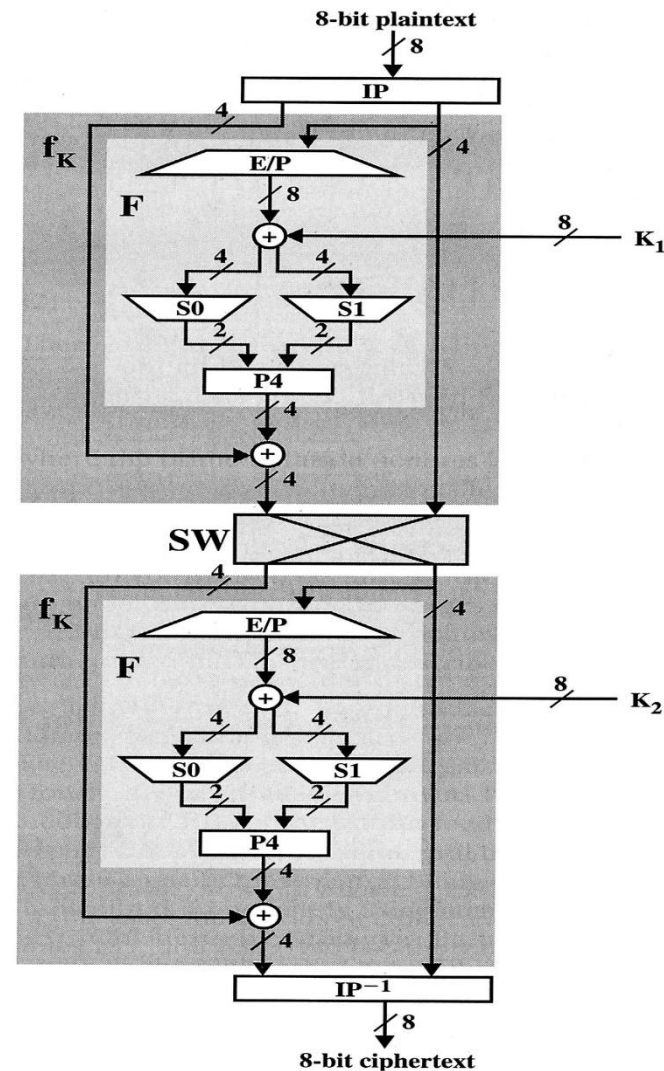
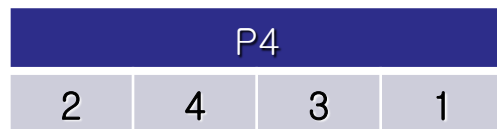
- E/P(확장/수열)



- S-box의 S0, S1

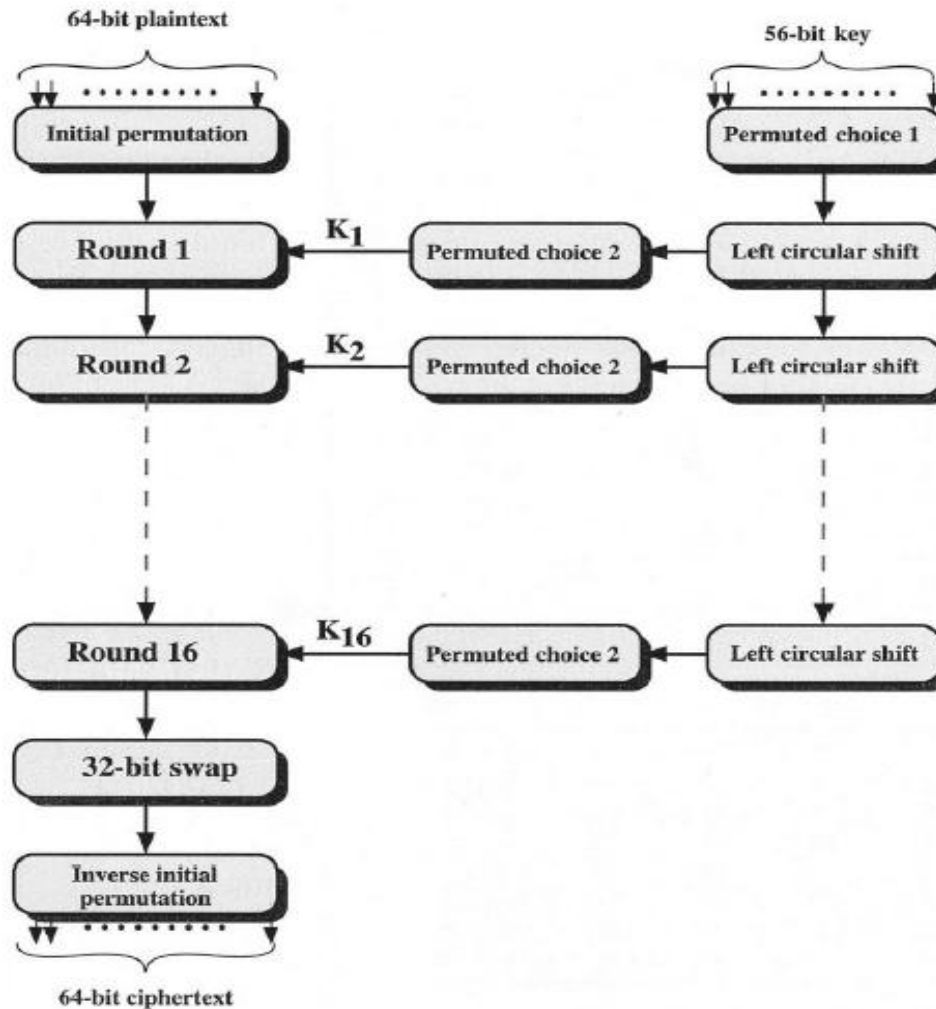
$$S_0 = \begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \end{matrix} \quad S_1 = \begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix} \end{matrix}$$

- P4



3.3 DES의 구조

- 암호화키 56비트를 이용하여 64비트 출력으로 변환



3.3 DES의 구조

- 초기순열(IP)

58	50	42	34	26	18	10	3
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	61	43	35	27	19	11	3
61	63	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- 역 초기순열(IP⁻¹)

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- 확장 순열(E)

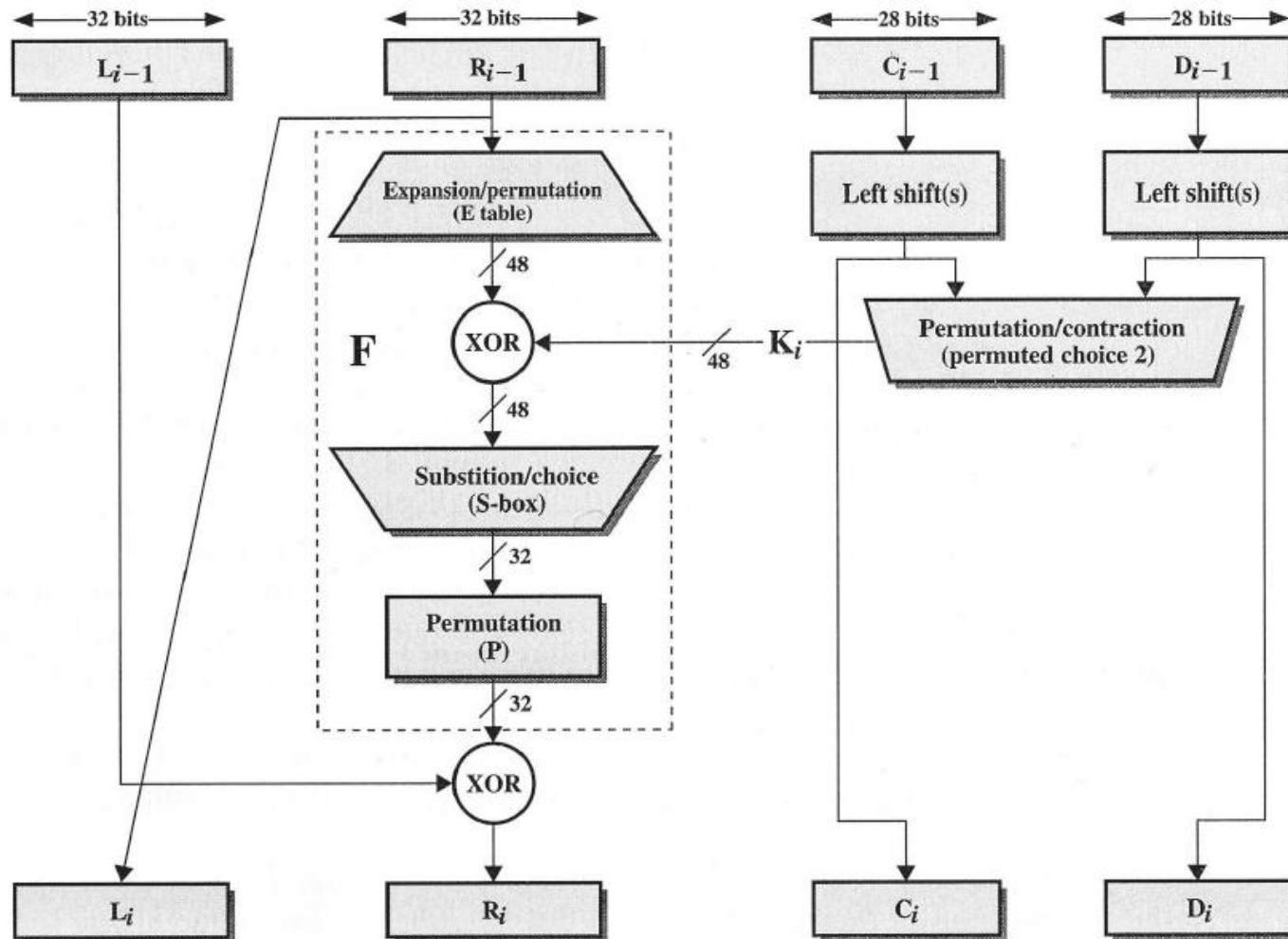
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- 순열함수(P)

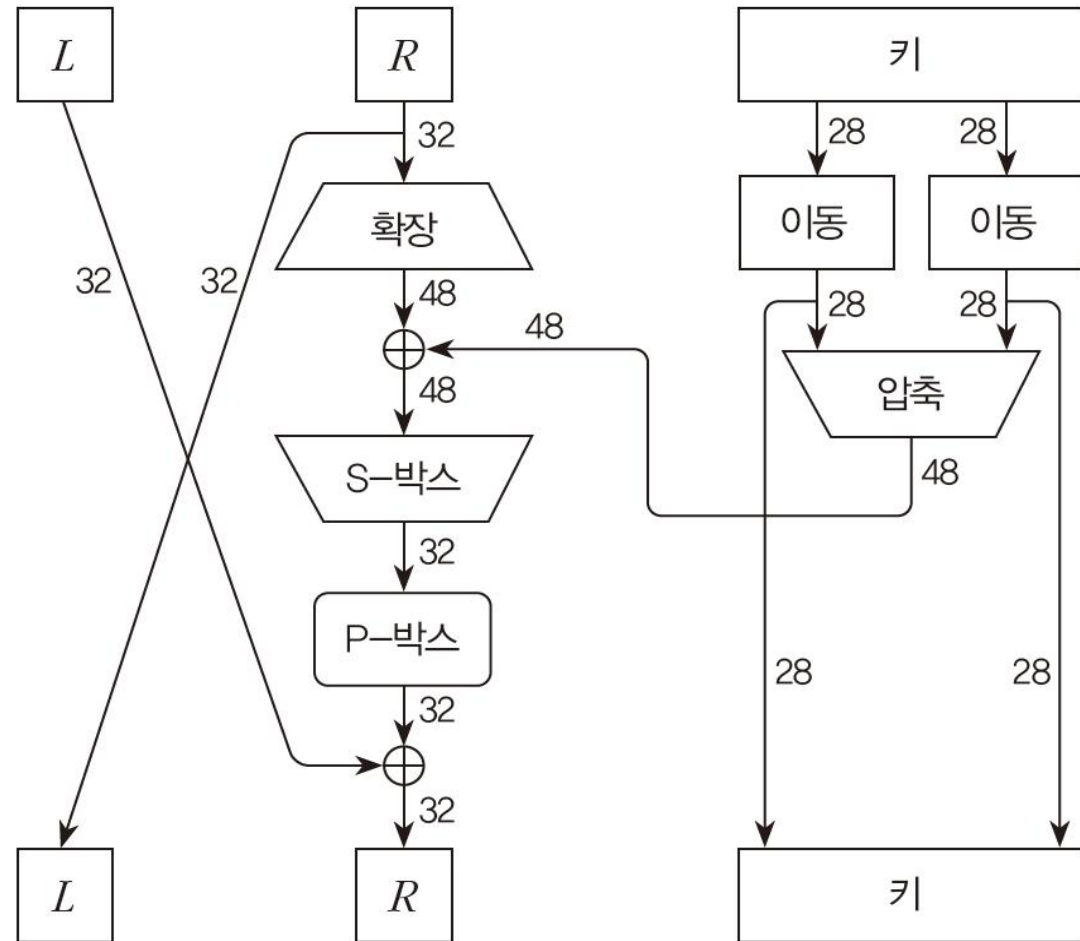
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	16	32	27	3	9
19	13	30	6	22	11	4	25

3.3 DES의 구조

- DES 알고리즘의 단일 반복 과정



■ DES 단일 회전



■ DES 키 스케줄 알고리즘

DES 키 스케줄 알고리즘

$i=1,2,\dots,n$ 회전 각각에 대하여

$LK=r_i$ 비트만큼 LK 를 왼쪽으로 시프트

$RK=r_i$ 비트만큼 RK 를 왼쪽으로 시프트

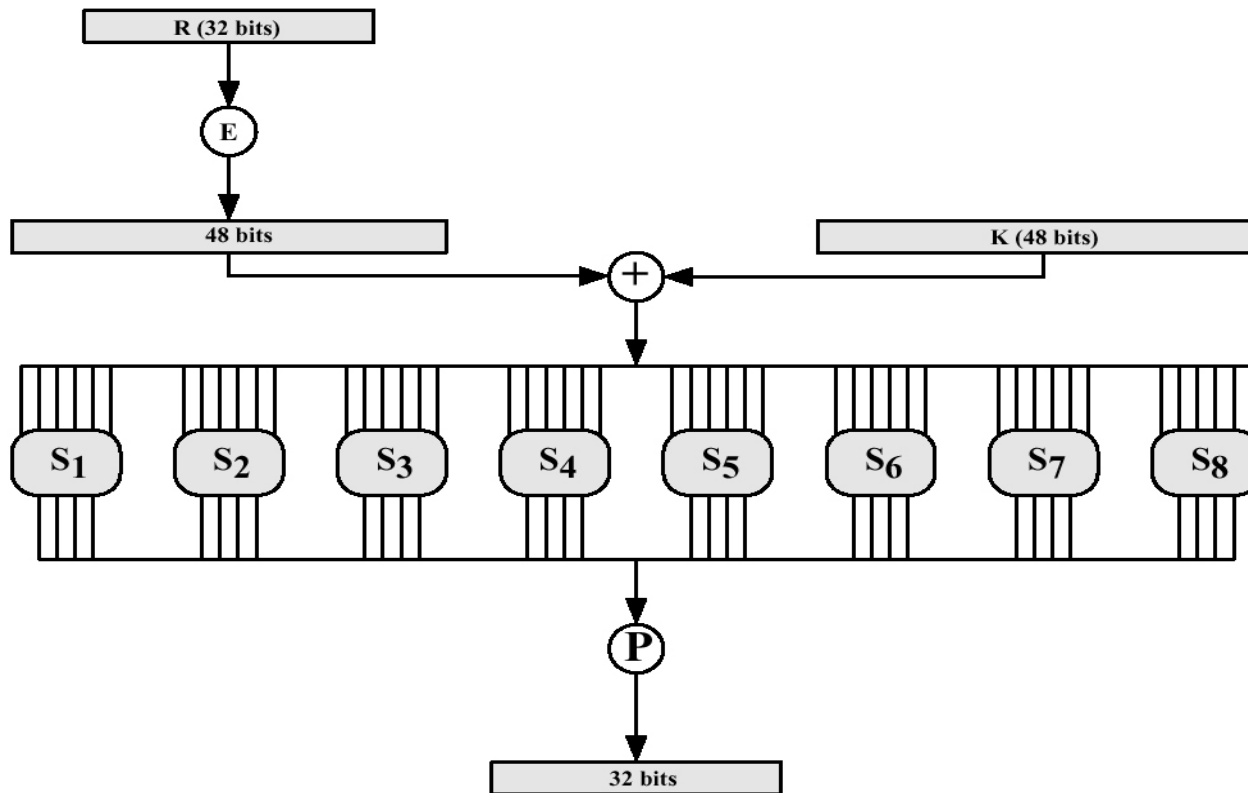
보조키 K_i 의 왼쪽 절반은 LK 의 LP 비트로 구성

보조키 K_i 의 오른쪽 절반은 RK 의 RP 비트로 구성

next i

3.3 DES의 구조

- 함수 $F(R, K)$ 의 계산
 - 8개의 S-box로 구성
 - 각 S-box는 6비트 입력, 4비트 출력 생성



- DES S-박스

DES S-박스 1(16진수)

$b_0 b_5$	$b_1 b_2 b_3 b_4$															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

- 예) S1박스 입력 6비트: 011011

행(01) 열(13)

출력 4비트: 5 → 0101

3.3 DES의 구조

- DES의 S-박스 정의

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4



7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

 S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

 S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

 S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

3.3 DES의 구조

- 키 생성

- DES 키의 단계별 계산표
- 순열선택1(PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	63	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

- 순열선택2(PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

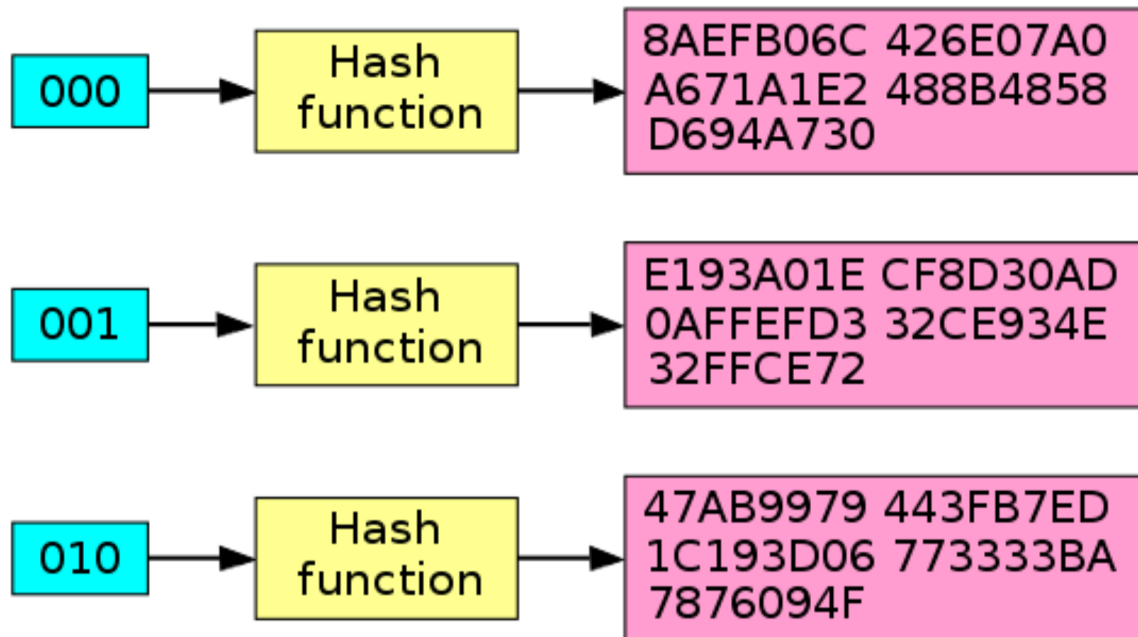
- 쇄도우효과 (Avalanche Effect)

평문이나 키의 작은 변화가 암호문에 대하여 대하여 중요한 변화를 일으키는 암호 알고리즘의 중요한 성질

- 예) 해시함수 SHA-1 : 1비트만 바뀌어도 결과 값이 완전히 달라짐

Input

Hash sum



3.3 DES 예제

- 예) DES 쇄도효과 : 평문변화
 - 평문이나 키를 1비트 바꿀 때 암호문의 여러 비트가 변함

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 acaca55ed16c3653	37
6	18b2fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682v2cefbcb	32
8	67117cf2c11bf609 2b2cefbcb99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP ⁻¹	da02ce3a89ecac3b 057cde97d7683f2a	32

DES 암호호화 원리 실전예제

3.5 차분 해독법과 선형 해독법: DES 공격법

차분 해독법 (Differential Cryptanalysis)

- 평문의 일부를 변경하면 암호문이 어떻게 변화하는지를 조사
- 블록암호를 보면 입력되는 평문이 1비트라도 달라지면 암호문은 전혀 다른 비트 패턴으로 변화
- 암호문의 변화들을 조사하여 해독의 실마리를 얻음

3.5 차분 해독법과 선형 해독법

선형 해독법 (Linear Cryptanalysis)

- 평문과 암호문 비트를 몇개 정도 XOR해서 0이 되는 확률을 조사
- 암호문이 충분히 랜덤하다면
 - 평문과 암호문의 비트를 몇개 XOR한 결과가 0될 확률은 $1/2$
- $1/2$ 로부터 크게 벗어난 비트의 갯수를 조사하여 키에 관한 정보를 얻음

3.5 차분 해독법과 선형 해독법

- DES는 두 공격법을 통해 해독 가능
- AES기반 현대 블록 암호는 차분/선형 해독법에 대해 안전하도록 설계

제4절 트리플 DES

4.1 트리플 DES란?

4.2 트리플 DES 암호화

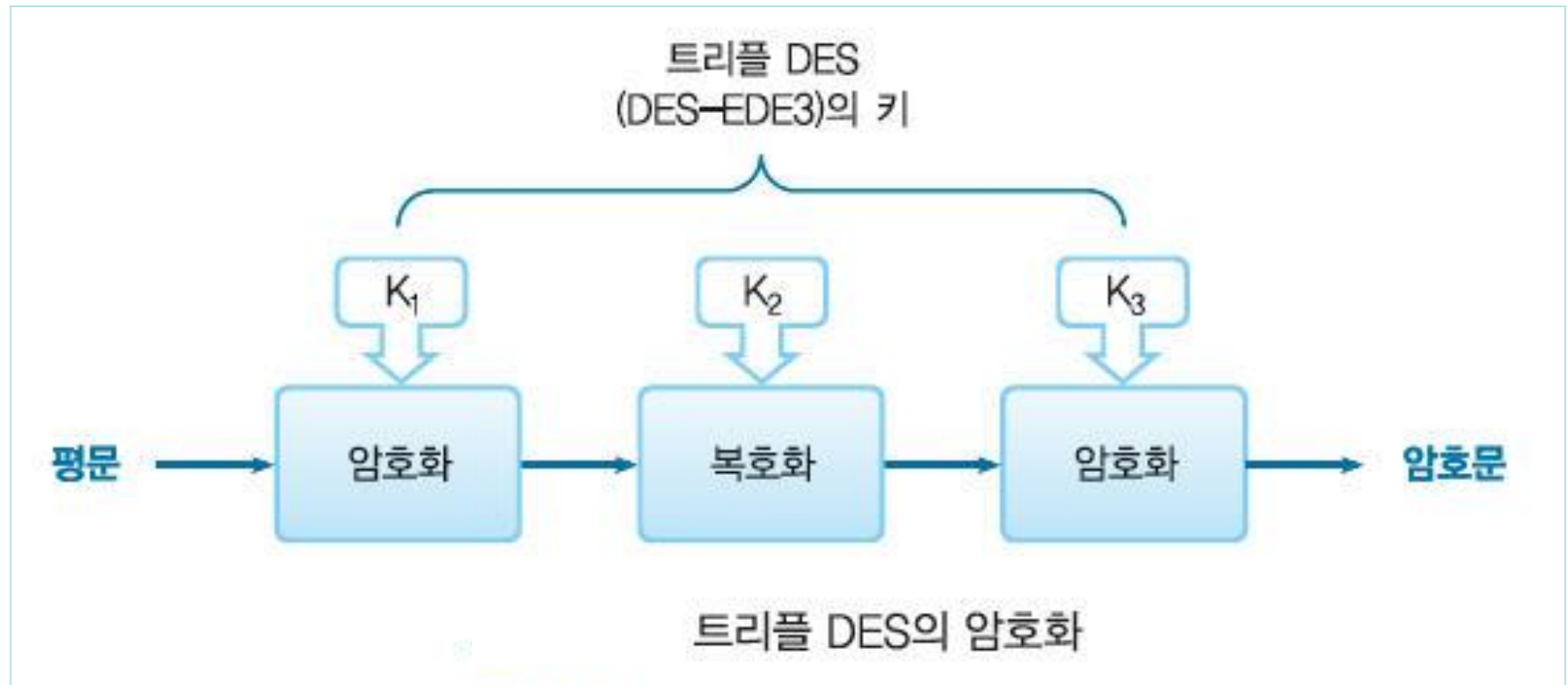
4.3 트리플 DES 복호화

4.4 트리플 DES의 현황

4.1 트리플 DES란?

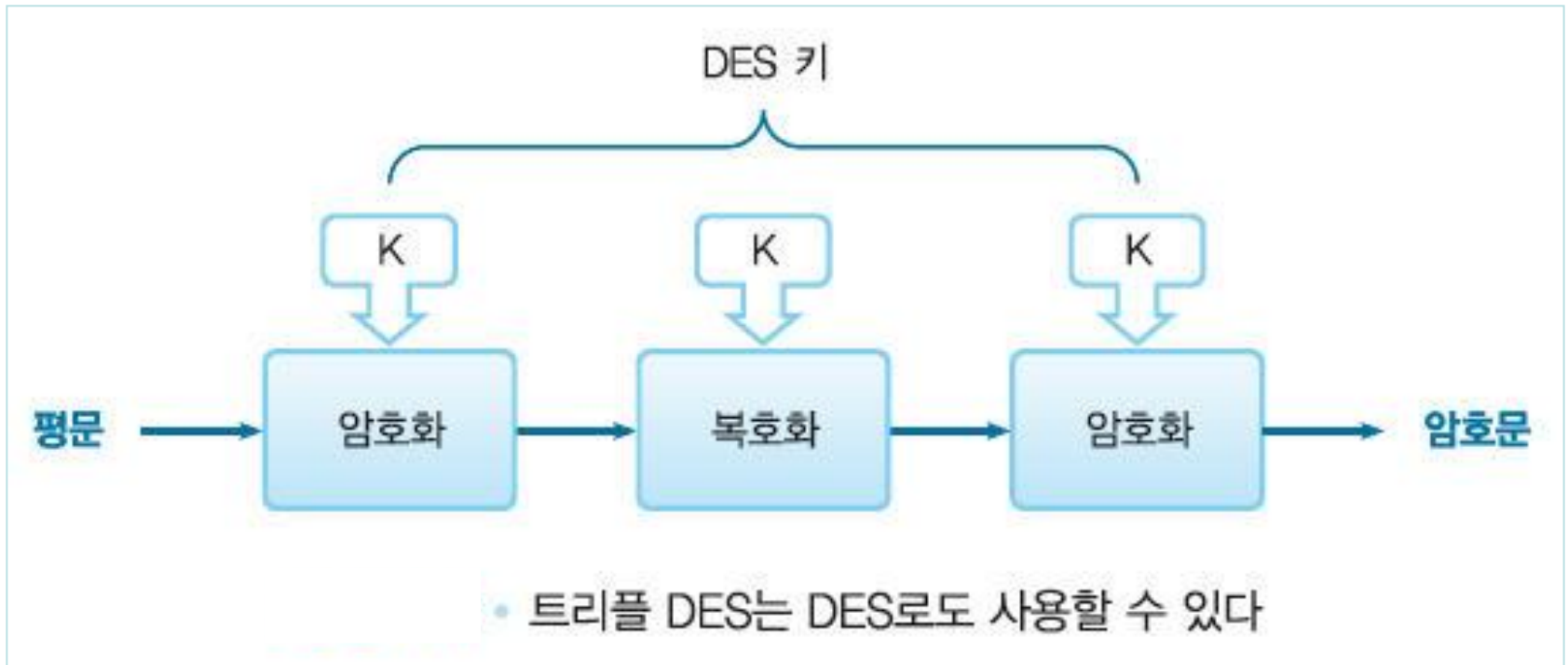
- 트리플 DES(Triple-DES)
- DES는 전사공격으로 현실적인 시간 내에 해독
- DES를 대신할 블록 암호가 필요
- 이를 위해 개발된 것이 트리플 DES
- DES보다 강력하도록 DES를 3단 겹치게 한 암호 알고리즘

4.2 트리플 DES 암호화



- 트리플 DES키: 168비트

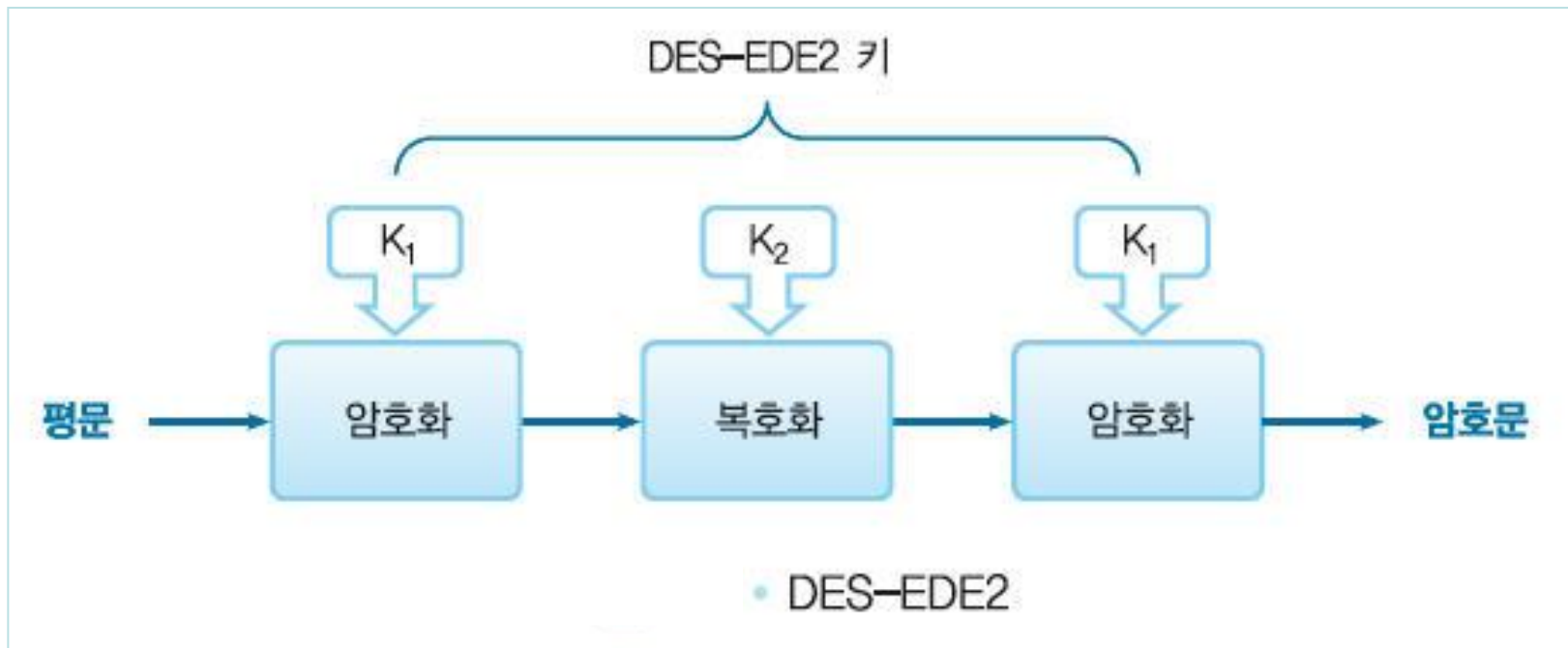
트리플 DES는 DES로도 사용



트리플 DES 종류

- DES
 - 모든 키에 같은 비트열을 사용
- DES-EDE2
 - 키1과 키3에 같은 키를 사용하고 키2에 다른 키를 사용
 - EDE는 암호화(Encryption) → 복호화(Decryption)
→ 암호화(Encryption) 순서
- DES-EDE3
 - 키1, 키2, 키3을 모두 다른 비트열을 사용

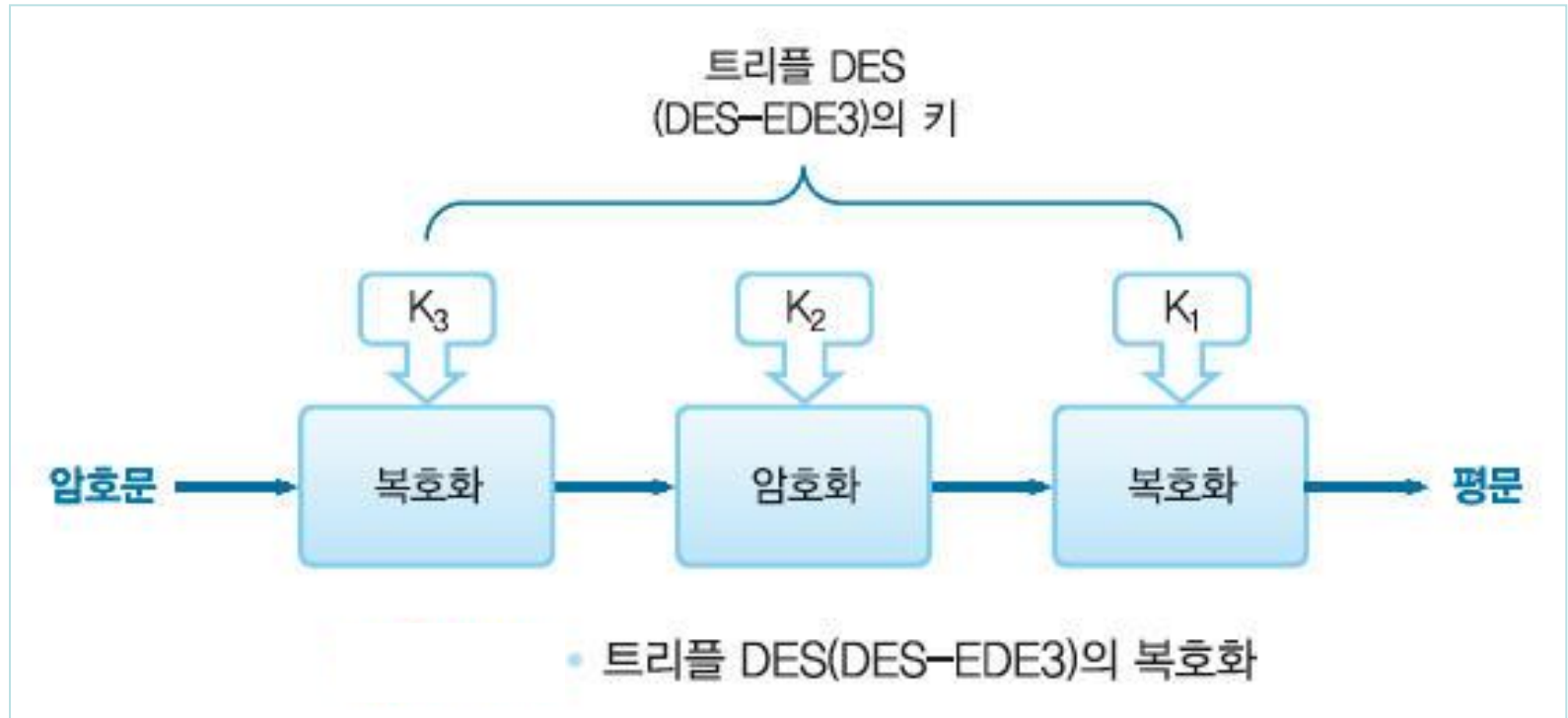
DES-EDE2



4.3 트리플 DES 복호화

- 암호화의 역순
- 키3, 키2, 키1의 순으로 복호화 → 암호화 → 복호화를 행한다

트리플 DES(DES-EDE3)의 복호화



4.4 트리플 DES의 현황

- 처리 속도는 빠르지 않음
- 안전성면에서도 풀려버린 사례가 있음
- 우리나라에서는 3-DES를 표준으로 정하지 않음
- 우리나라 국가표준은 SEED 및 ARIA

제5절 AES 선정 과정

5.1 AES란?

5.2 AES 선정 과정

5.3 AES 최종 후보 및 선정

5.1 AES란?

- AES (Advanced Encryption Standard)
 - DES를 대신한 새로운 표준 대칭 암호 알고리즘
 - AES의 후보로서 다수의 대칭 암호 알고리즘을 제안했지만, 그 중에서 **Rijndael**이라는 대칭 암호 알고리즘이 2000년에 AES로서 선정

5.2 AES 선정 과정

- NIST(National Institute of Standard and Technology)에서 공모
- 경쟁방식에 의한 표준화(standardization by competition)
- 조건
 - 제한 없이 무료로 이용
 - ANSI C와 Java에 의한 구현
 - 암호해독에 대한 강도의 평가
 - 암호 알고리즘 설계 규격과 프로그램 공개

5.3 AES 최종 후보 및 선정

- 1차 심사 통과: 15개
 - CAST256, Crypton, DEAL, DFC, E2, Frog, HPC, LOKI97, Magenta, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish
- 2차 심사 통과: 5개

명칭	응모자
MARS	IBM
RC6	RSA
Rijndael	Daemen, Rijmen
Serpent	Anderson, Biham, Knudsen
Twofish	Counterpane

제6절 Rijndael

6.1 Rijndael이란?

6.2 Rijndael의 암호화와 복호화

6.3 Rijndael의 해독

6.4 어떤 암호를 사용하면 좋은가?

6.1 Rijndael이란?

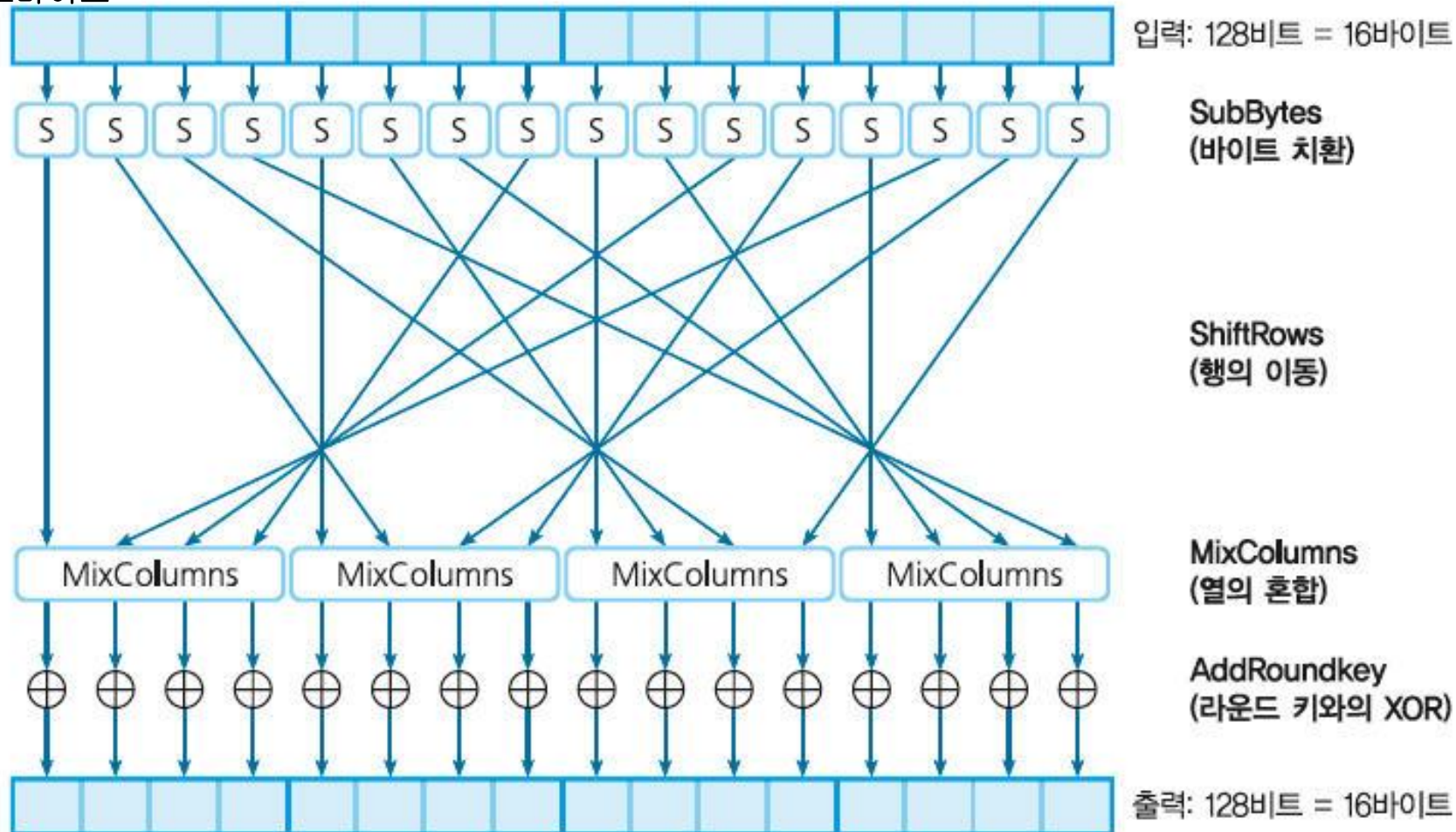
- 벨기에 연구자 Joan Daemen과 Vincent Rijmen이 설계한 블록 암호 알고리즘
- 블록 길이
 - 128비트
- 키의 비트 길이
 - 128비트
 - 192비트
 - 256비트

6.2 Rijndael의 암호화와 복호화

- 복수의 라운드(round)로 구성(10~14)
- SPN(Substitution-Permutation Network) 구조
(Feistel 구조가 아님)
- SubBytes
- ShiftRows
- MixColumns
- AddRoundKey

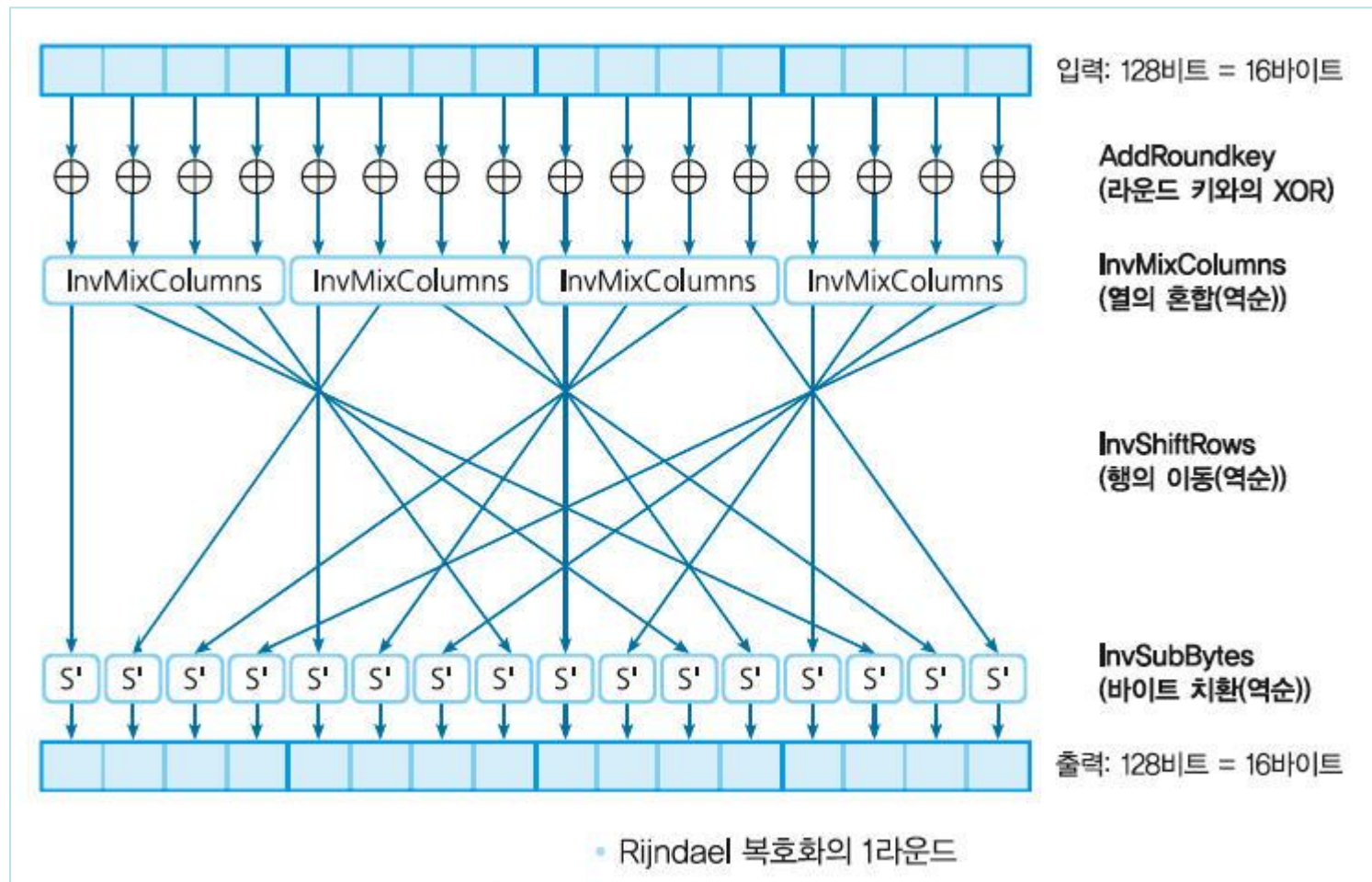
Rijndael 암호화 1라운드

1바이트



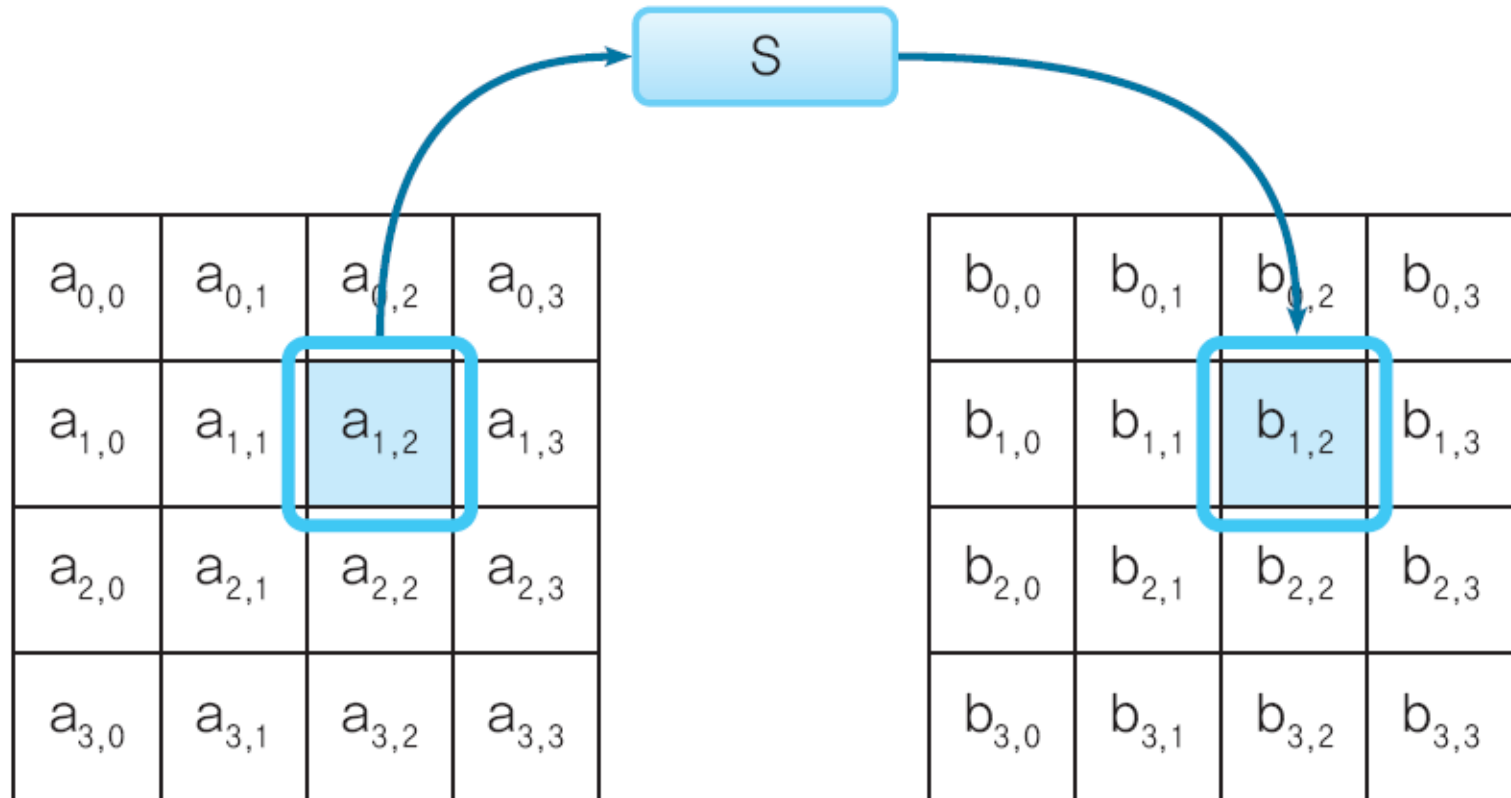
Rijndael 암호화의 1라운드

Rijndael 복호화 1라운드



SubByte(바이트 대체)

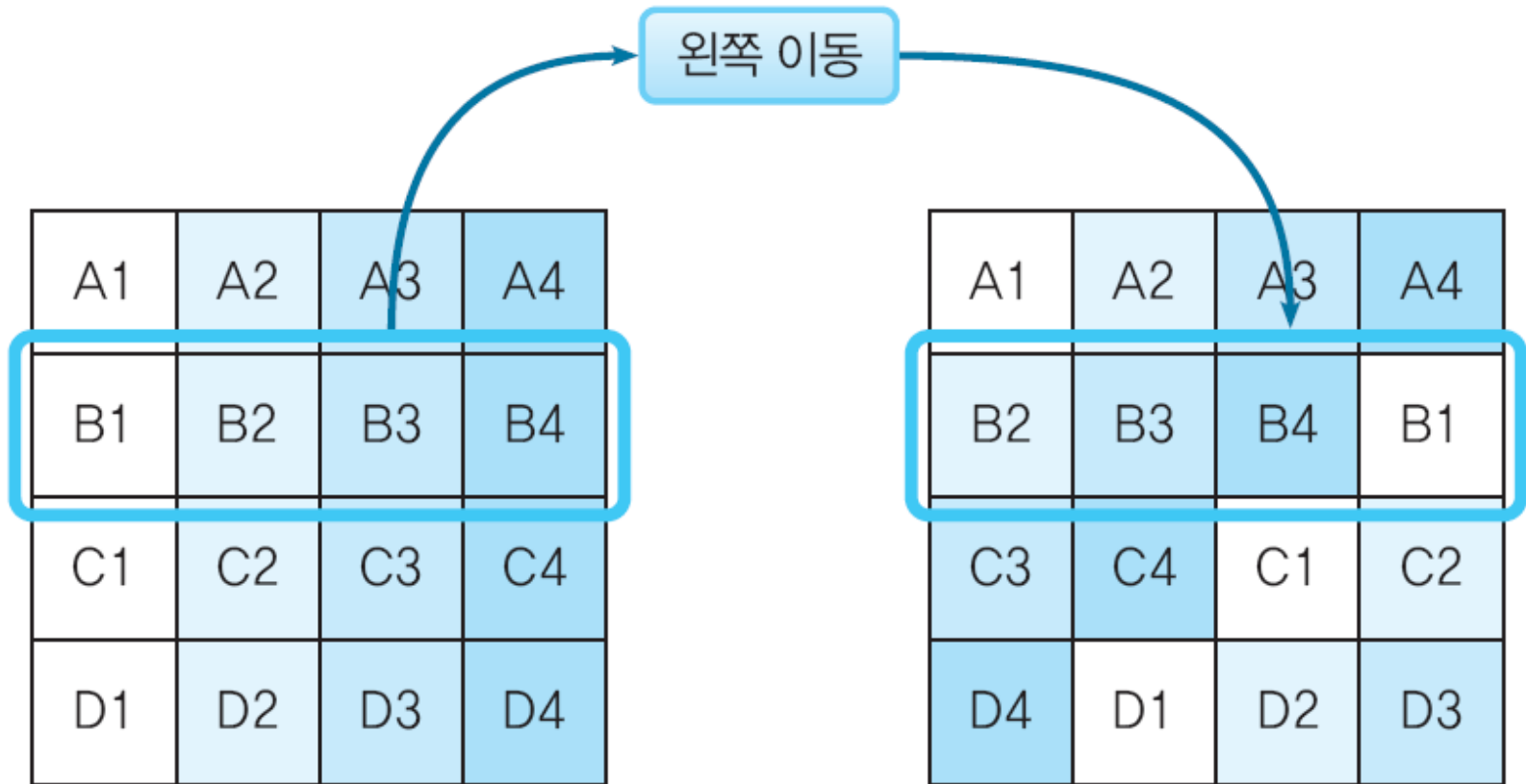
- SubByte처리는 1바이트(0~255중 어떤값)을 인덱스로 하고, 256개의 값을 가지고 치환표(S박스)로부터 1개의 값을 얻음
- 16바이트중 S박스를 통해 1바이트만 변환 하는것과 같음



- SubBytes(바이트 대체)

ShiftRows(행 이동)

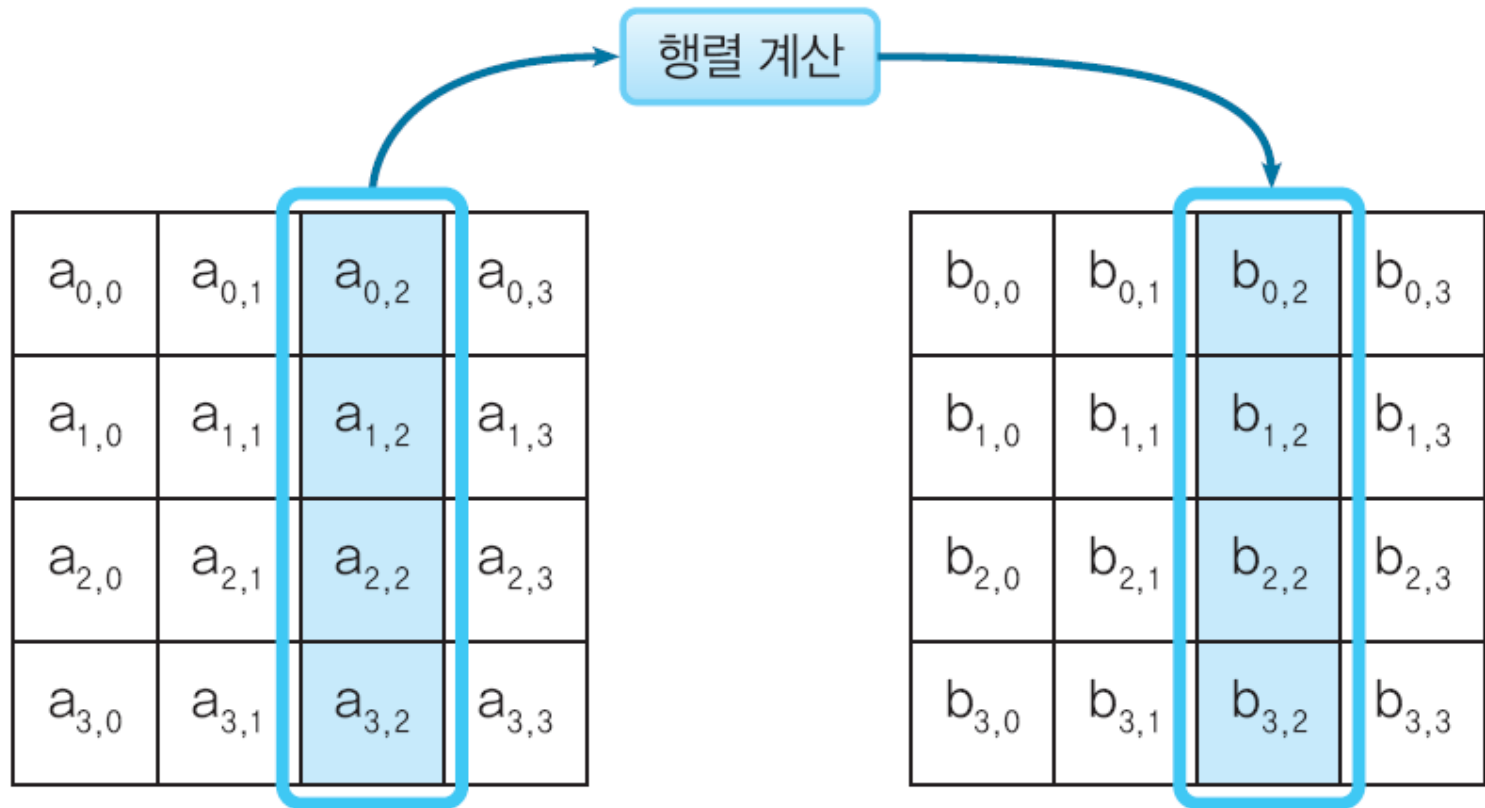
- 4바이트 단위로 왼쪽이동해서 썩는 과정



- ShiftRows(행 이동)

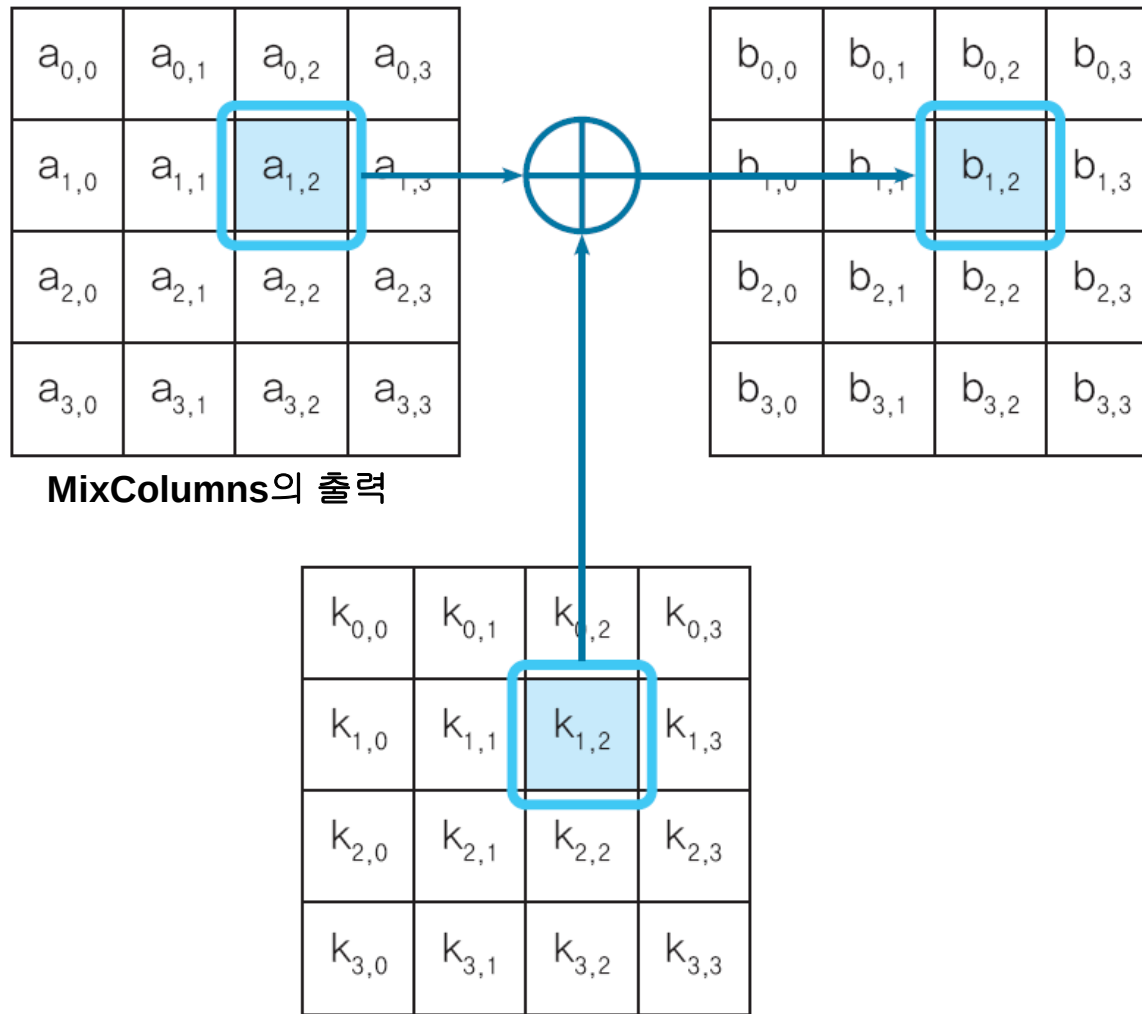
MixColumns(열 섞기)

- 4바이트 값을 비트연산을 써서 다른 4바이트 값으로 변환



- MixColumns(열 섞기)

AddRoundKey(라운드 키와 XOR)



- AddRoundKey(라운드 키와 XOR)

- Rijndael 암호화 1라운드

- SubBytes → ShiftRows → MixColumns → AddRoundKey

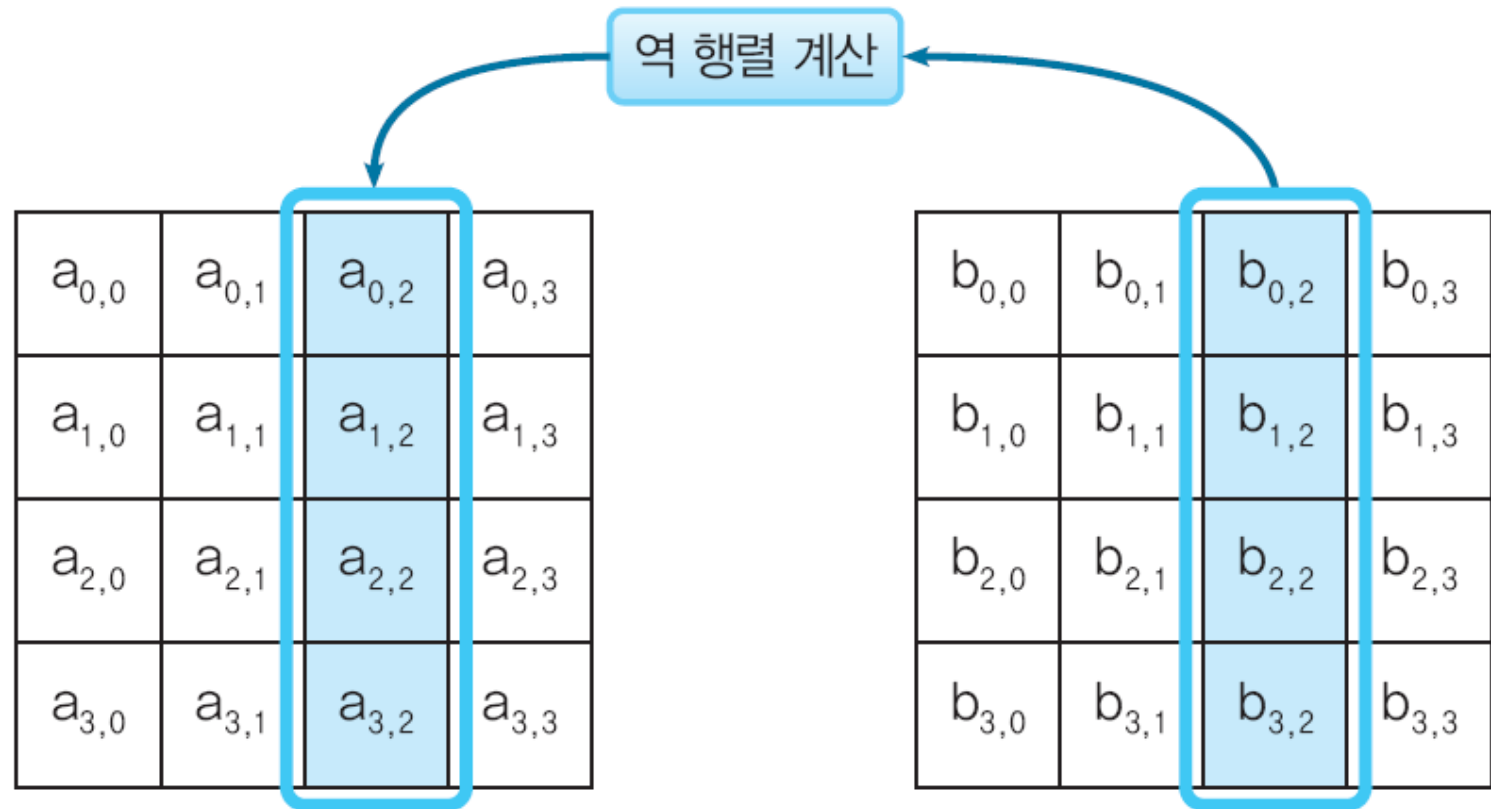
- 복호화 (역순)

- AddRoundKey → InvMixColumns → InvShiftRows →
InvSubBytes

- AddRoundKey는 동일하게 라운드 키와 XOR

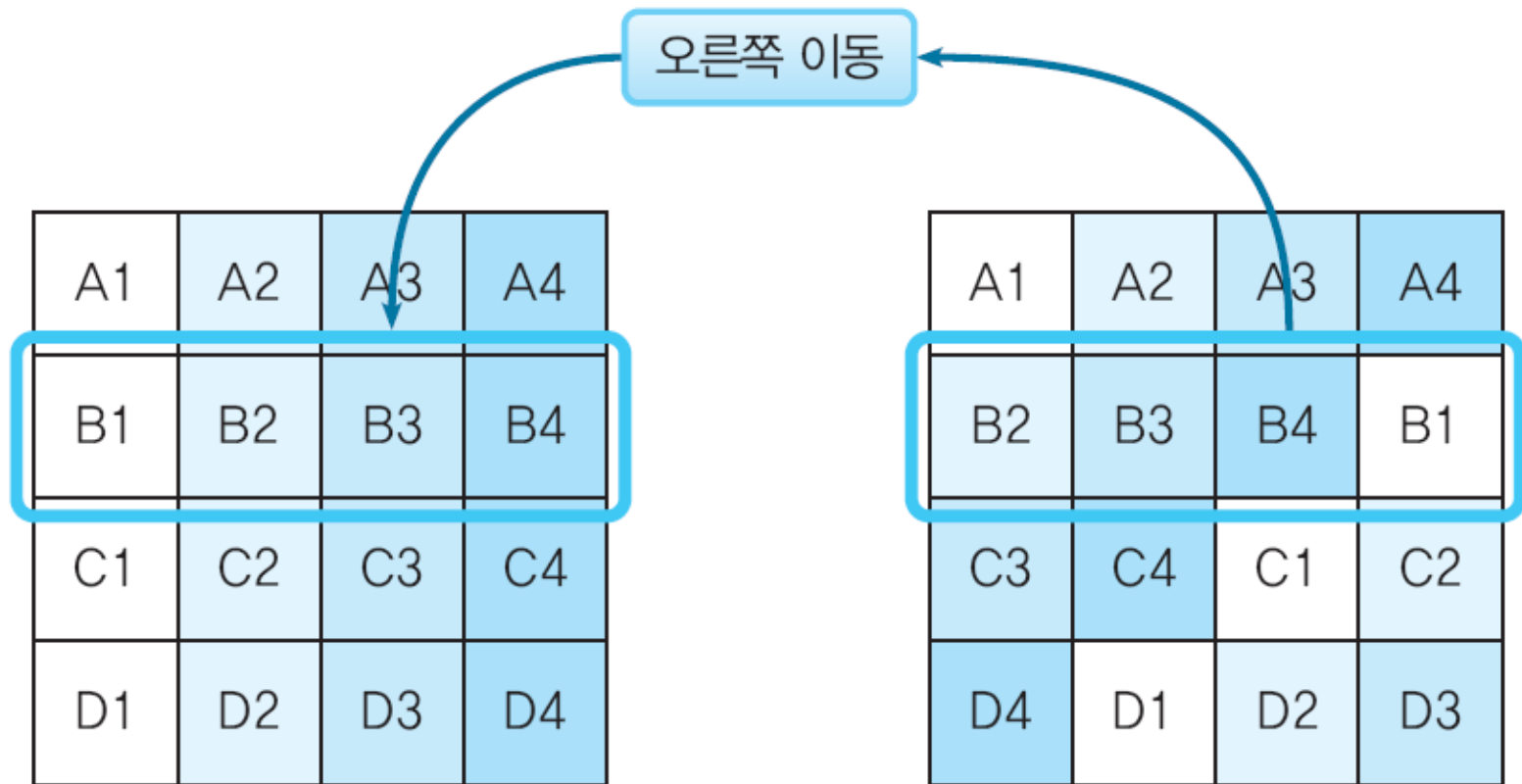
- 다른 연산은 역처리(Inv~~)

InvMixColumns(역 열 섞기)



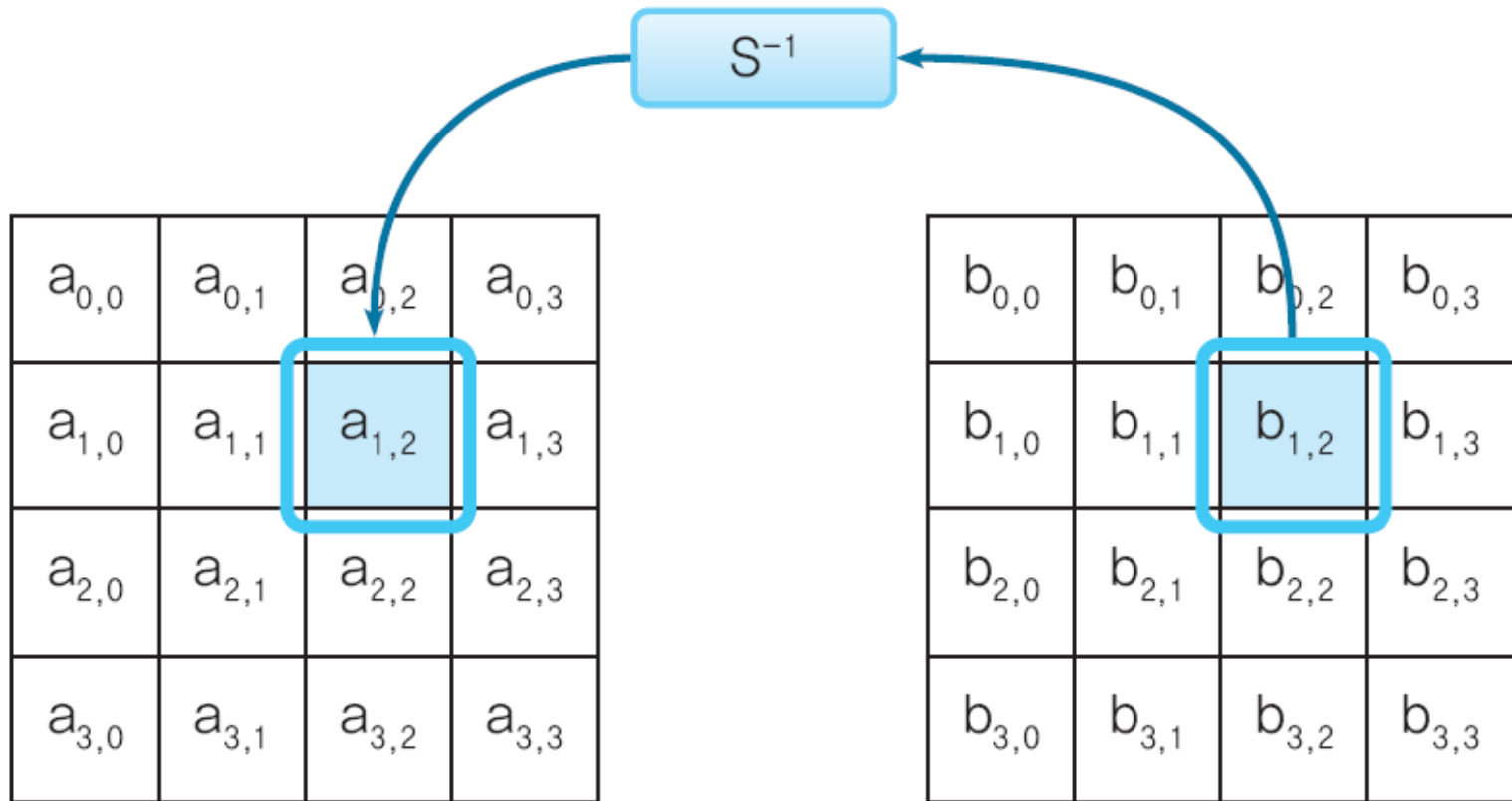
- InvMixColumns(역 열 섞기)

InvShiftRows(역 행 이동)



- InvShiftRows(역 행 이동)

InvSubBytes(역 바이트 대체)



- InvSubBytes(역 바이트 대체)

6.3 Rijndael의 해독

- Rijndael 알고리즘의 수학적 구조
 - Rijndael의 수식을 수학적인 조작에 의해 풀 수 있다면, Rijndael을 수학적으로 해독할 수 있을 것이다
- Rijndael에 대한 유효한 공격은 현재로서는 발견되지 않음

6.4 어떤 암호를 사용하면 좋은가?

- DES
 - 사용하지 말것
 - 과거 소프트웨어와의 호환성 유지를 위해 필요
- 트리플 DES
 - 안전성 및 호환성 때문에 미사용이 권고됨
 - AES로 대체
- AES(Rijndael)
 - 고속
 - 다양한 플랫폼
 - 현재 까지 안전
 - 사용 권장
 - AES 최종 후보 5개도 사용가능

기타 대칭키 암호 알고리즘

- SEED

- ✓ 한국 인터넷진흥원과 국내 암호전문가들이 함께 개발한 알고리즘
- ✓ 128 비트 블록 암호, 128 비트 비밀키, 64 비트 라운드 키를 생성, 총 16회 라운드 진행
- ✓ 차후 256 비트의 비밀키를 사용하는 SEED-256이 개발

- ARIA(Academy Research Institute Agency)

- ✓ 128 비트의 블록 암호, Involutional SPN 구조
- ✓ 128, 192, 256 비트의 비밀키 사용
- ✓ 입출력 크기와 사용할 수 있는 비밀키의 크기는 AES와 동일

- HIGHT(High security and light weight)

- ✓ 저전력 및 경량화 컴퓨팅 환경의 기밀성 제공을 위해 개발된 64 비트 블록 암호

- LEA(Light weight Encryption Algorithm)

- ✓ 국가보안 기술 연구소가 개발한 128 비트 경량 고속 블록 암호
- ✓ 대용량 데이터를 빠르게 처리하거나 스마트폰 보안, 사물인터넷(IoT) 등 저전력 암호화에 널리 사용

기타 대칭키 암호 알고리즘

- IDEA(International Data Encryption Algorithm)
 - ✓ 스위스 연방 기술 기관에서 개발
 - ✓ 128 비트 키, 64 비트 블록 암호로 Feistel과 SPN의 중간 형태
- RC5
 - ✓ 미국 Rivest가 개발
 - ✓ 비교적 간단한 연산으로 빠른 암호/복호화 제공 (DES보다 10배 빠름)
 - ✓ 32, 64, 128 비트 블록 암호, 0 ~ 2040 비트 키, 라운드 수: 1 ~ 255 회 가변적

- NIST(미국 국립표준기술연구소)는 양자컴퓨터 시대에 대비하여 양자 내성암호(PQC, Post-Quantum Cryptography) 표준화 진행
 - 주로 [공개키 암호에](#) 초점
 - 대칭키 암호에 대해서는 약간 다른 접근이 필요

Shor 알고리즘 vs Grover 알고리즘 비교표

항목	Shor 알고리즘	Grover 알고리즘
제안자 / 연도	Peter Shor (1994)	Lov Grover (1996)
영향을 받는 암호	공개키 암호 (RSA, ECC 등)	대칭키 암호 (AES, 해시 등)
기술적 특징	소인수분해, 이산 로그를 양자 효율로 계산	전체 탐색(brute-force)을 $\sqrt{N}$ 으로 단축
암호에 미치는 영향	기존 공개키 암호 완전 파괴	대칭키 암호 보안 강도 절반 수준으로 약화
대응 방법	새로운 공개키 암호 (PQC) 필요	키 길이 증가 (예: AES-256 사용)
보안성 대응 여부	기존 알고리즘 사용 불가	기존 알고리즘 사용 가능 (단, 보완 필요)

NIST - 양자 컴퓨터 대비 대칭키 암호 알고리즘 동향

- 양자 컴퓨터가 대칭키 암호에 미치는 영향
 - Grover 알고리즘: 대칭키 암호의 보안 강도를 제곱근 수준으로 감소
 - 예: AES-128 → 양자 환경에서 보안 강도 약 64비트로 하락
 - 하지만 완전한 파괴 아님, 대응 가능
- NIST의 대응 전략
 - 기존 대칭키 알고리즘 유지, 다만 키 길이 증가 권장
 - AES-256, SHA-3 계열 → 양자 안전성 확보 가능
 - 대칭키 관련 양자 공격 모델에 대한 이론 분석 및 키 관리 지침(SP 800 시리즈) 제 공 (업데이트 중)
- 권장되는 대칭키 알고리즘

알고리즘	상태	양자 대비 보안성
AES-128	사용 가능	보안성 낮음 (권장 안함)
AES-256	표준	안전 (Grover 대응)
SHA-2	표준	SHA-512 이상 권장
SHA-3	표준	안전
KMAC, cSHAKE	SHA-3 기반 MAC/해시	안전

Q1. 대칭 암호에 대한 설명 중 잘못 된 것은?

- ① 암호화와 복호화에 동일한 키를 사용한다.
- ② 처리 속도가 비대칭 암호에 비해 빠르다.
- ③ 키의 길이에 따라 안전성이 달라진다.
- ④ 키 배송 문제를 해결하기 쉽다.
- ⑤ 예로 AES가 있다.

Q2. 다음 설명 중 틀린 것은? (복수 가능)

- ① 블록 암호와 스트림 암호 모두 평문을 처리할 때 바이트 단위로 처리한다.
- ② 스트림 암호에서는 의사 난수를 사용한다.
- ③ 블록 암호에서는 키의 길이가 블록의 길이와 동일하다.
- ④ 블록 암호는 대칭 암호와 비대칭 암호 모두에 적용이 가능하다.
- ⑤ 스트림 암호에서 사용하는 것은 XOR이다.

Q3. DES에 대한 설명 중 올바른 것은?

- ① 대칭 암호와 비대칭 암호 모두에 사용된다.
- ② 블록의 길이가 56비트이다.
- ③ 14 라운드로 이루어졌다.
- ④ 평문블록은 제일 먼저 비트 열의 순서를 재배열하는 초기치환을 거친다.
- ⑤ S-상자는 48비트를 입력으로 받아들여 48비트를 출력한다.

Q4. AES에 대한 설명 중 옳바르지 않은 것은?

- ① DES를 대신하는 새로운 표준이다.
- ② 블록 길이가 128 비트인 대칭 블록 암호이다.
- ③ 키의 길이에 따라 라운드 수가 달라진다.
- ④ 키의 길이는 128, 192, 와 256 비트를 지원 한다.
- ⑤ 페이스텔 구조를 기반으로 하여 작성되었다.

Q5. AES 선정 요구조건에 대한 설명 중 옳바르지 않은 것은?

- ① 선정 후 무료로 사용하는 것을 전제로 하였다.
- ② 암호화 자체의 속도뿐만 아니라 키의 셋업 속도도 중요하다.
- ③ 블록의 길이는 128 비트로 요구하였다.
- ④ 고정된 키의 길이를 요구하였다.
- ⑤ 플랫폼에 무관하게 효율적으로 동작되어야 한다.

Q6. 트리플 DES에 대한 설명 중 옳바르지 않은 것은?

- ① DES의 암호화를 3번 되풀이하고 있는 것은 아니다.
- ② 트리플 DES의 처리 속도는 AES의 처리속도보다 빠르다.
- ③ 트리플 DES를 이용해 DES로 암호화된 것을 복호화 할 수 있다.
- ④ 트리플 DES의 키는 궁극적으로 $56 \times 3 = 168$ 비트가 된다.
- ⑤ 트리플 DES는 DES보다 보안 강도가 높다.

Q7. Rijndael에서 입력되는 각 바이트에 대한 4 가지 처리에 해당되지 않는 것은?

- ① ReplaceBytes.
- ② SubBytes.
- ③ ShiftRows.
- ④ MixColumns.
- ⑤ AddRoundKey.

Q8. 다음 설명 중 옳바르지 않은 것은?

- ① AES는 키의 길이에 따라 10~14 라운드로 구성된다.
- ② 비트 열의 치환과 대체가 대칭 암호 알고리즘의 근본이다.
- ③ 스트림 암호에서는 입력되는 요소의 합이 일정한 블록의 크기가 된 후에 처리한다.
- ④ 같은 비트 열로 한 비트 열에 XOR을 2회 되풀이하면 원래 비트 열로 돌아간다.
- ⑤ 의사난수 생성기에서는 초기값을 사용하는데 초기값이 동일하면 동일한 의사난수를 출력한다.

Q9. 다음 XOR 계산 결과 중 잘못된 것은?

① $a \oplus b = b \oplus a$

② $a \oplus b \oplus b = a$

③ $a \oplus a = 0$

④ $a \oplus 0 = a$

⑤ $1 \oplus a = a$

Q10. 1회용 패드는 실제로 왜 사용되지 않는지 그 이유로 적합하지 않은 것은?

- ① 키 배송 문제가 해결되지 않는다.
- ② 키를 보존하는 문제가 생긴다.
- ③ 키에 사용하는 난수열은 한 번 밖에 사용할 수 없으므로 난수를 대량으로 생성해야 한다.
- ④ 송신자와 수신자 사이의 통신에 있어서 동기화 문제가 있다.
- ⑤ 키를 생성할 때 의사난수 생성기를 활용해야 하는 불편함이 있다.

참고문헌

- Cryptography and Network Security, William Stallings, Pearson
- 컴퓨터보안과 암호, William Stallings (최용락 외 2명 옮김), 그린출판
- 위키백과, 파이스텔 암호
- 정보보안 이론과 실제 (3판), 김경곤 외1, IT CookBook
- DES Animation, Youtube, <https://www.youtube.com/watch?v=Vcld7CMAnNs>
- Shor, P. W. (1997). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 26(5), 1484–1509
- Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96), 212–219.

Q & A

Thanks!